1971

A compiler for the interactive solution of differential equations.

Nelson, Harvey Gordon.

Monterey, California ; Naval Postgraduate School

http://hdl.handle.net/10945/15614

A COMPILER FOR THE INTERACTIVE SOLUTION OF DIFFERENTIAL EQUATIONS

HARVEY GORDON NELSON

United States Naval Postgraduate School

THESIS

A COMPILER FOR THE INTERACTIVE SOLUTION OF DIFFERENTIAL EQUATIONS

by

Harvey Gordon Nelson

Thesis Advisor: G. J. Thaler

June 1971

A Compiler for the Interactive
Solution of Differential Equations

by

Harvey Gordon Nelson
Lieutenant, United States Navy
B.A., Vanderbilt University, 1963

Submitted in partial fulfillment of the
requirements for the degree of

ELECTRICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
June 1971

ABSTRACT

A digital simulation language for the interactive definition and solution of piece-wise continuous non-linear ordinary differential equations using the Runge-Kutta method has been designed and implemented. The combination of interactive graphics approach and a special differential equation description language make the analysis program very versatile and easy to use. For second order systems, a grid of phase trajectory segments over the user specified phase plane is used as a background for the solutions.

TABLE OF CONTENTS

LIST OF DRAWINGS

I. INTRODUCTION

The goal of this study was to develop an interactive analysis tool for the study of piecewise continuous non-linear ordinary differential equations. It is recognized that there are available in most computer centers several programs for solving non-linear differential equations. However they are either very special purpose or else require that the problem and/or control description be inserted into a subroutine or part of the analysis program. This in turn must be compiled by the computer before execution time. If the user wants to change some feature of the system, he must change the applicable program cards and recompile. Thus in general only one problem type per computer run is possible.

Also there exists a large number of continuous digital simulation languages [Clancy and Fineberg, 1965]. These are best represented at this time [Cardenas and Karplus, 1970] by MIMIC, DSL/90, and CSMP. DSL/90 and CSMP, which is based on DSL/90, have translators which convert the prescribed input language into FORTRAN subroutines. These subroutines are then compiled by the FORTRAN compiler and executed with the solution routine. This is a multi-step process which is both time consuming and not compatible with the interactive approach. CSMP and DSL/90 will accept the problem description either in the block diagram form or the differential equation form. There is a version of CSMP (similar to the above CSMP in name only) which uses the block diagram approach only and is suitable for use on the small computers of the IBM line. MIMIC

incorporates its own compiler and has a very flexible algebraic capability.

The program described in this thesis uses the differential equation approach and has a built-in compiler flexibility similar to MIMIC. In addition it has been designed from its conception to be fully interactive using graphics as the interface between the user and the program. All problem description is handled as data. Thus only the compiled version of the analysis program is needed by the user. This is accomplished by the use of a special differential equation description language. An automatic compiler (transparent to the user) built into the program provides the interface between the description language and the fourth order Runge-Kutta solver. This use of a special differential equation description language and the interactive graphics approach give the program considerable flexibility in the type of differential equations it can handle.

The class of piece-wise continuous non-linear ordinary differential equations handled by this program is restricted by the following considerations:

1. The equation must have only one independent variable.

2. The equation must be normalized so that the coefficient and exponent of the highest order derivative are both one.

3. At each given moment during the solution one and only one differential equation can be applicable. The applicable differential equation must be well defined for the conditions at that point. Note that this allows piece-wise continuous differential equations with discontinuities but excludes simultaneous differential equations.

These are the only restrictions inherent in the method of solution. The implementation described in this thesis has been arbitrarily limited to differential equations of eighth order or less. For a given problem there may be added restrictions in defining the differential equations due to the limitations of the program language used to define the systems and/or the ingenuity of the user. However, if the user can define his system using the defined programming language and the above three requirements are not violated then the program will solve the system.

Although the language used in the system description is quite natural to most users, it is important to completely specify it. Thus Chapter III deals with a description of the elements of the language and the principles of its use.

Other features of the program are:

1. The program has the generality required to be useful for a wide variety of applications. In addition, the program does not require sophisticated computer techniques on the part of the user.

2. The outputs of the program are a smooth x vs x' plot (phase plane in the second order case) of the solution and printer plots of the time response of each state variable.

3. The "window" represented by the x vs x' plane can be adjusted in size, scale, and position. These three parameters can be adjusted in the x and x' directions independently; thus the user has full flexibility to adjust the window as desired.

4. The solution is accomplished using the state variable approach with fourth order Runge-Kutta integration. The parameters of the solution are completely under the control of the user.

5. For second order time independent systems there is superimposed on the phase plane a grid of phase trajectory segments. These slope markers represent the slope of the trajectory if it were to pass through that area of the phase plane. Thus without obtaining a solution one can visually obtain a feel for what the system will do for various initial conditions. This in turn aids the user in deciding what the essential characteristics of the system are.

6. For the Interactive Graphics Version, the parameters of each feature mentioned above are available for modification as desired on an interactive basis. For example one may select specific values of various parameters and observe their effect on the resultant solution. The program is interactive and prompts the user in the choice of responses indicating his desires.

There are two versions of this program. One version is called the Batch Version and runs on the IBM 360/67. The other is an Interactive Graphics Version and is implemented on the Electrical Engineering Department's XDS 9300 with the associated ADAGE Graphics Terminals. The interactive graphics version was primarily practical due to the power of the "intelligent" Adage Terminals.

In Chapter II a general description of the program and its various parts are discussed. A major portion of the coding is the compiler and interpreter which are one of the primary features of this program.

Chapter IV concerns itself with a description of the program from the users viewpoint. In addition to the general comments, there are included sections which deal separately with the Batch Version and the Interactive Graphics Version.

The use of the Interactive Graphics Version requires a knowledge of how to use the graphics system. This is explained by a suggested first session exercise. In this exercise the user is led step-by-step through all the techniques required to use the program.

Chapter V ties all the above together by presenting a large number of actual problems that have been studied with this program. The input deck is explicitly presented along with the resultant output.

A special benefit of the analysis feature of this program is its usefulness as a teaching or laboratory tool. This is especially true for the Interactive Graphics Version used to analyse a second order time independent system. For example the user could type in the description for a simple second order pendulum system. The slope lines give a very dramatic overall view of the effect of various types and amounts of damping. Since the system is interactive, the user can immediately pursue and verify theoretical observations.

II. GENERAL DESCRIPTION OF THE PROGRAM

A. INTRODUCTION

The program consists of five primary parts. The first four are the compiler, interpreter, slope line generator (used in the second order time independent case only), and the solution generator. Each of these will be explained in more detail later in this chapter. For the present, a brief functional description will be given.

The "Compiler" takes the differential equation or system of differential equations, as the case may be, and through a process to be described later comes up with a polish string [Forsythe, 1969, Burroughs, 1964] of integer codes stored as a single vector. This polish string is the compiled version of the input system of equations.

The "Interpreter," when directed, uses the polish string and the current value of all its required arguments and produces a value for the derivative of each state of the system with respect to time. The required arguments are indicated by the system equations and may include the current values of the states, time, and the values assigned the constant coefficients A through H.

The resultant state variable derivatives as defined by the interpreter are needed by both the slope line generator and the solution generator. The slope line generator calculates the grid of slope lines as shown in Chapter V. The solution generator, as the name implies, generates the time solution.

The fifth primary part of the program is the coordinator. It is the coordinator that ties the above together and makes it work. It

provides the required scheduling based on the results of the user's modifications. These modifications are accomplished on an interactive basis when using the XDS/AGT Interactive Graphics Version and via data cards on the IBM 360/67 Batch Version.

B. MATHEMATICAL BASIS OF THE PROGRAM

1. The General Mathematical Concept

Assume an n^{th} order differential equation of the form

$$\frac{d^n x}{dt^n} + f(x) = g(x, u) \tag{2-1}$$

where f(x) is a function of the n-1 derivatives, and g(x, u) is a function of the inputs and the n-1 derivatives. This equation can be rewritten as n first order differential equations. Let us assume the following notation and substitutions

$$\begin{aligned}
x &= Z(1) \\
x' &= Z(2) = ZDOT(1) \\
x'' &= Z(3) = ZDOT(2) \\
x''' &= Z(4) = ZDOT(3) \\
&\vdots \\
x^{(n-1)} &= Z(n) = ZDOT(n-1).
\end{aligned} \tag{2-2}$$

Rewriting equation 2-1 using the substitutions and notation specified in Equations 2-2 gives

$$\begin{aligned}
ZDOT(1) &= Z(2) \\
ZDOT(2) &= Z(3) \\
&\vdots
\end{aligned} \tag{2-3}$$

$$ZDOT(n-1) = Z(n)$$

$$ZDOT(n) = g(x, u) - f(x).$$

Each of the Z's will be called the state variables of the system. Note that for the first n-1 state variables the derivative of the i^{th} state variable of the system is equal to the i+1 state variable. The derivative of the n^{th} state variable is equal to g(x, u) - f(x).

The solution generator needs only these derivatives of the state variables of the system. The compiler and interpreter use the above observations as the basis for the method of implementation.

Assume that equation (2-1) has been presented to the system. The first action of the compiler is to recognize the term $\frac{d^n x}{dt^n}$ which is denoted on the data card as an "x" followed by n apostrophes. This notation is completely analagous to the use of dots to denote time derivatives. The notation will be explained in detail in Chapter III. Having recognized this term it now knows the order of the differential equation. Next the expression g(x, u) - f(x) is compiled. The compiled version will be used by the interpreter to evaluate the desired derivatives.

The interpreter first evaluates the compiled version of g(x, u) - f(x) which becomes the value of the highest order state derivative. The other derivatives are defined using the state values as defined in Eq. (2-3).

2. <u>Mathematical Derivation of the Slope Lines</u>

The slope lines are used only for second order time independent systems. Thus the comments of this section will apply to a second order system only. The general system can be written as

$$x'' + f(x) = g(x) \tag{2-4}$$

where $f(x) + g(x,u)$ are as defined in equation 2-1.

Letting $x = Z(1)$ and $x' = Z(2)$ gives

$$\begin{aligned} dx/dt &= ZDOT(1) = Z(2) \\ dx'/dt &= ZDOT(2) = g(x,u) - f(x). \end{aligned} \tag{2-5}$$

For a given time t the slope of the solution trajectory is

$$\begin{aligned} \text{slope} &= (dx/dt)/(dx'/dt) \\ &= ZDOT(2)/Z(2). \end{aligned} \tag{2-6}$$

Thus knowing a position and a corresponding slope the desired lines can be generated using simple trigonometry.

C. COORDINATOR

Figure 2-1 is a simplified block diagram of the program and outlines the function of the coordinator. The coordinator provides the overall coordination and interface for obtaining the desired chain of events. The program is best summarized by reviewing this block diagram.

The first action of the program is to obtain the system and plot description data. This includes the system equations, the plot parameters such as scales and sizes, and the solution information such as the initial conditions, initial and final times. With this information a complete run through the program can be made.

Next the compiler compiles the system description data. The results of this compilation, if successful, are saved in a vector (a 1-dimensional array or stack). If the result of the compilation or one of the steps which uses the compiled version encounters an illegal situation then an appropriate message is presented and the program

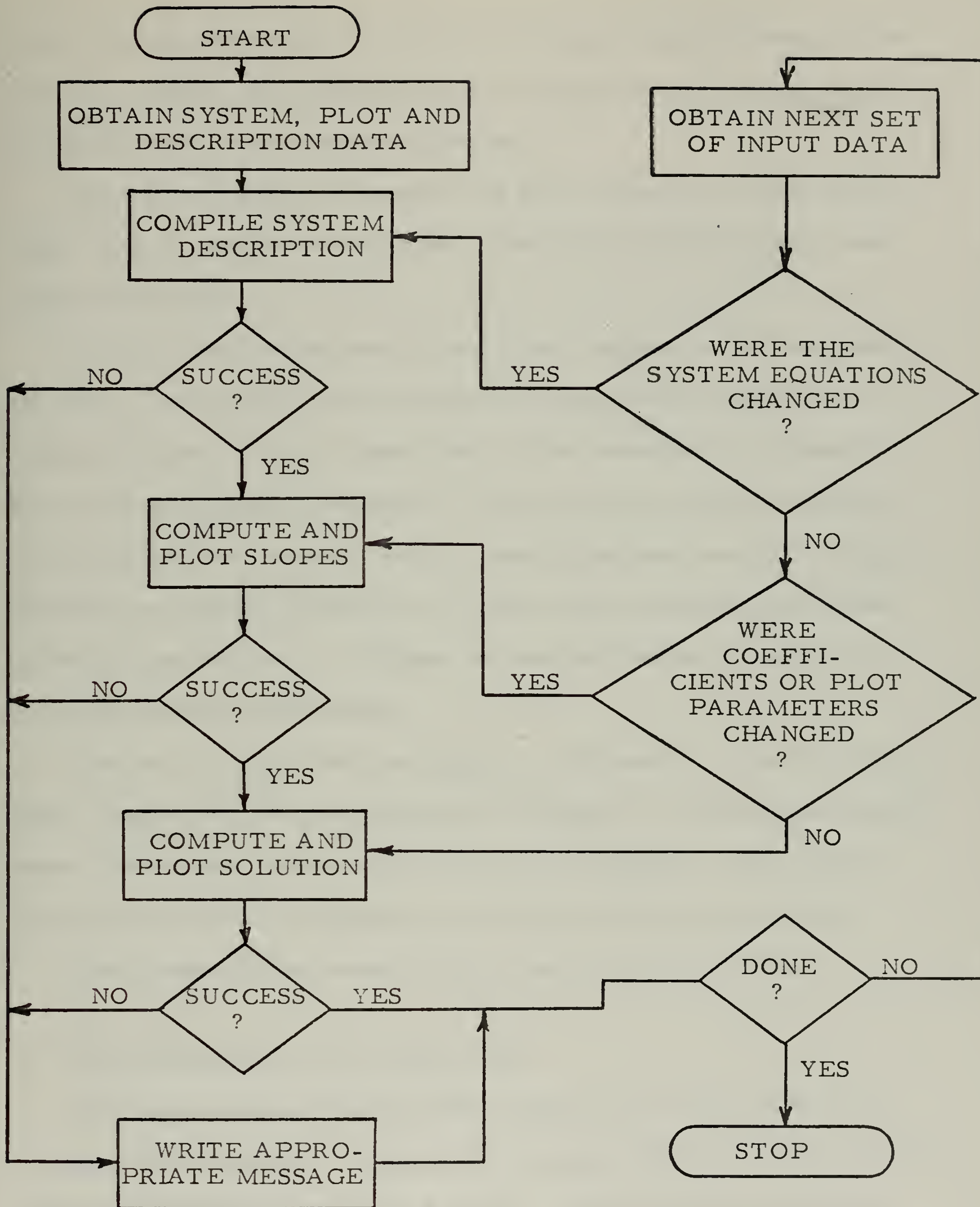

Fig. 2-1 OVERALL BLOCK DIAGRAM

goes on to get new data. For the Batch Version the next new problem would be started. For the Interactive Version the user would make the appropriate corrections and proceed.

After a successful compilation the plot is started and the grid of slope lines is computed and plotted. The time solution is then computed and plotted.

At this time, the program is ready for the next set of data from the user. Depending on what new data is obtained several courses of action are taken. If the system description equations are changed then a recompilation is required. The program flow after recompilation is the same as before. If the system equations were not changed, then the coordinator checks for a change in the constant coefficients or the plot parameters. A change in these will cause a transfer to the slope generating segment.

The Batch Version differs slightly at this point. Instead of asking if the "coefficients or plot parameters changed?," the question should read: "Is the next solution desired on a new graph?" This allows a family of solutions to be plotted on the same graph if so desired.

The above cycle of events continue until the user is done.

D. DESCRIPTION OF THE COMPILER

The programming details of the compiler are not of central importance to this thesis. A copy of the coding is in the overall program printout included as Programs A and B. Frequent use of comment statements including explanation of codes, key variables, and such are included.

Referring to the simplified block diagram, fig. 2-2, a very brief description of the compiler will be given here. The compiler accepts

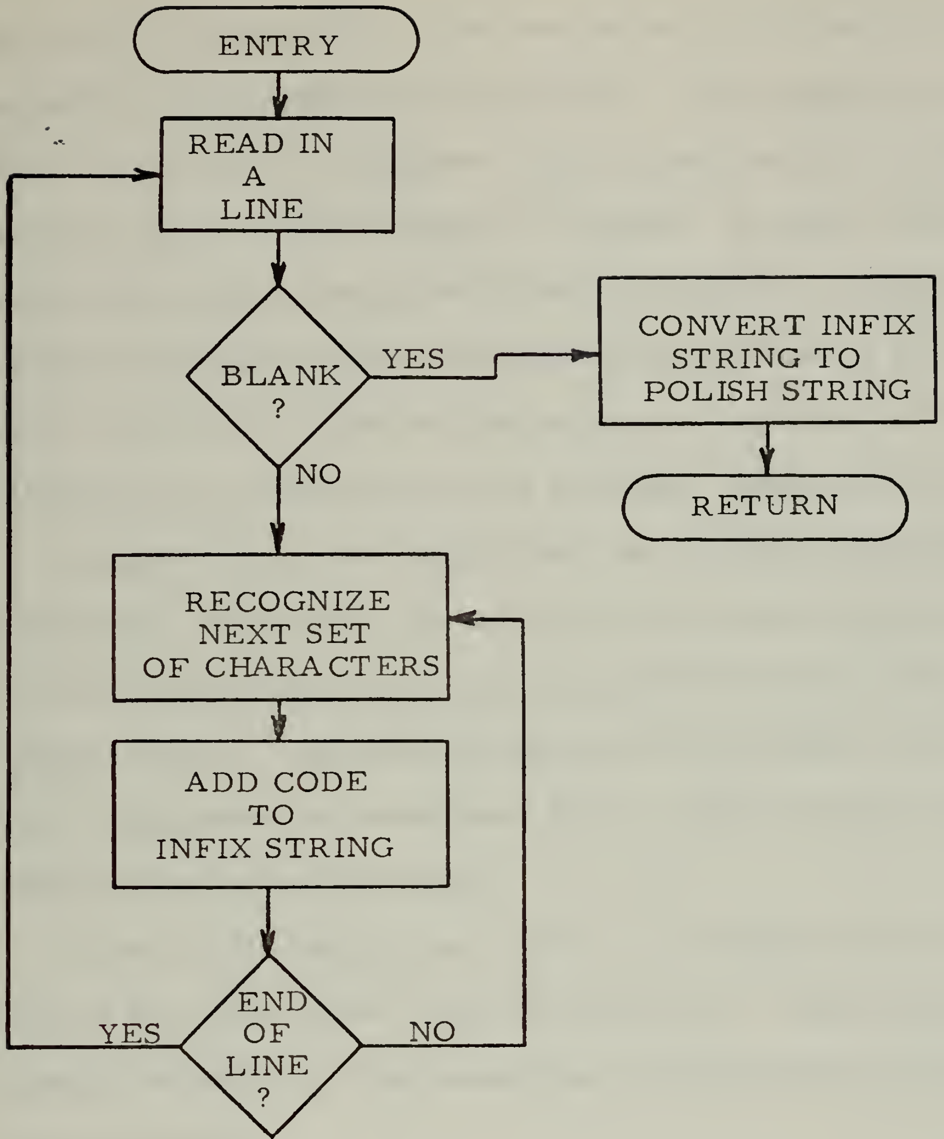

Fig. 2-2 COMPILER SIMPLIFIED BLOCK DIAGRAM

the input a line at a time. The recognition of the elements of the language is accomplished by brute force. For example, let us assume that a "T" has been recognized. This T could be the variable T or the first letter of either "TAN" or "THEN". A check of the following characters quickly reveals which of the three it is. When a language element is recognized a corresponding integer code is placed in a stack. At the end of the recognition phase, the entire set of equations will have been coded into a string of integer codes stored in the stack.

A blank card or line signals that the recognition phase has been completed. At this time the string is in the form known as "infix." The final task of the compiler is to convert this infix string to a "polish" string. The polish string has the advantage that it is quite easy to implement an interpreter for it. This interpreter will be discussed in the next section.

A part of the compilation work is to recognize and use information such as the order of the differential equation. The order of the differential equation is determined from the equations and is required by the interpreter.

E. DESCRIPTION OF THE INTERPRETER

The purpose of the Interpreter is to take the compiled code which was generated in the compiler and values for the required operands which were specified by the user and to provide the derivatives that are needed by the slope and solution generators. The use of these derivatives will be explained in the next two sections.

The functional block diagram is presented in figure 2-3. Upon entry to the Interpreter all the required operands have been defined. These are the present values of each of the state variables, the values

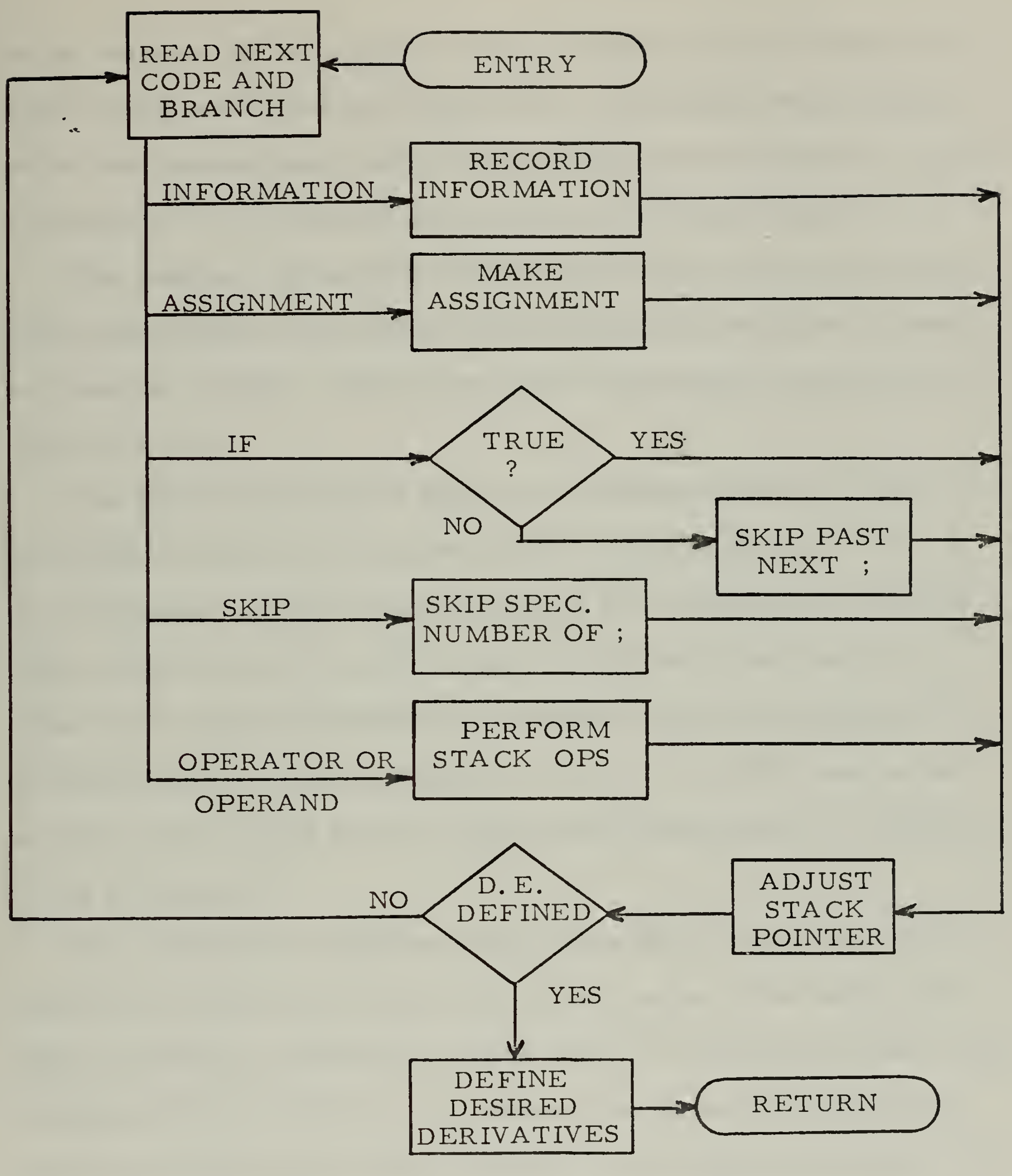

Fig. 2-3 INTERPRETER SIMPLIFIED BLOCK DIAGRAM

of the constants (A through H), and the problem time T. The Interpreter takes one by one each coded entry in the polish stack (result of the compilation phase) and carries out the required operation. The various types of actions are summarized by the block diagram.

The code may be carrying information such as the order of the differential equation and whether the differential equation to be used is a function of time. These codes cause the proper variables and flags to be set.

The logical branching is done in the following manner. Each statement in the stack is separated by a code representing a ";". If the interpreter evaluates the argument of an IF statement to be false, then it skips through the code to the entry following the next ";". Thus it has skipped the part of the statement indicating action to be accomplished if the logical argument is true. The SKIP instruction uses the same method but will skip several instructions as requested by the programmer.

The arithmetic evaluation of the polish stack is accomplished by the use of software LIFO (last in first out) stacks. The basic principles involved in evaluating a polish stack are well known [Rosen 1967, Forsythe 1969]. For ease in implementation three different LIFO stacks have been used. One is used for the arithmetic operations, one for logical operations, and one to hold addresses. Thus they are respectively real, logical, and integer stacks. Associated with these stacks there are the required pointers to keep track of their top positions.

When the interpreter notes that a differential equation has been evaluated it terminates the interpretation phase. As will be explained

in Chapters III and IV the program uses the first differential equation whose conditions for use are met. There it will be shown that the program easily handles a system whose behavior must be described by different differential equations in different portions of the solution space. The last action of the Interpreter is to define the desired system derivatives as described in Section B. These derivatives plus the order of the system are passed back to the calling slope or solution generator.

F. DESCRIPTION OF THE SLOPE GENERATOR

The Slope Generator presents a grid of slope markers over the area of the phase plane. The use of these slopes is demonstrated in Chapter V. The mathematical basis of the slopes was described in Section B of this chapter.

For each grid point on the phase plane the interpreter is called. If the system, for that grid point, is not second order or time independent then it skips on to the next grid point. The effect of this skip is to suppress the slope markers at that point. If the system is of second order and not time dependent then the slope is defined and is drawn at the current grid point.

The resultant derivative information from the interpreter and the position information are used to calculate the desired slope. The slope is used in turn to compute the vector that represents the slope at that point. This derivation is made somewhat more complicated by the scaling options available to the user. Thus both the slopes and the vector length calculation must consider the horizontal and vertical scale factors.

G. DESCRIPTION OF THE SOLUTION GENERATOR

The Solution Generator incorporates a very straightforward use of the Runge-Kutta method. The heart of the Runge-Kutta solution is the function subroutine RKLDEQ. This function subroutine is part of the IBM/360 library. For flexibility the actual cards are included in this program.

After each solution point is generated the result is stored for future plots. In addition, the interactive graphics version presents the solution as it takes place. The solution rate for a typical system is about ten iterations per second with graphics. Thus the resultant presentation is approximately real time for a solution time step of 0.1 seconds.

III. GENERAL DESCRIPTION OF THE LANGUAGE

A. INTRODUCTION TO THE LANGUAGE

The language primarily resembles the familiar (to most) form of FORTRAN. Anyone familiar with one of the languages; FORTRAN, Algol, Basic, or PL/I will find the language used here quite easy. The requirements of this language are very limited and of a special nature. Thus, since we have a special purpose language, it is not complicated by the generality required of most languages.

B. ELEMENTS OF THE LANGUAGE

1. The Character Set

The following characters are used in this language.

a. The alphabetic characters as applicable.

b. The numeric characters, 0 through 9.

c. The special characters:

*
=
;
.
(
)
+
-
/
'
blanks

d. Blanks are optional and may be used as desired for clarity and neatness except in the middle of a word, variable name, or number. The semicolon is used as a delimiter allowing more than one statement per line and may optionally be added at the end of a line.

2. Constants

A constant is a fixed, unvarying quantity. The constants used in this program are of two general types. They can either be self-defining numeric values appearing in the source statements, or constants defined externally to the program.

There are eight externally defined constants with names A, B, C, ... H. The values of these real constants are defined by the user externally to the system statements. Thus the values of these constants may be changed as desired, between executions, without recompilation of the systems equations. The method of defining these constants is explained in Chapter IV.

The self-defined constants are recognized by the compiler during compilation. With the exception of integer powers all self-defined constants are treated internally as real. However, when used in the program, the use of a decimal point is optional. That is, the decimal point may be either explicit or implicit.

In the case of exponentiation, an integer, positive or negative, used as a power will yield the expected integer power results. On the other hand, non-integer powers are computed using the natural logarithm/exponentiation method.

Examples:

Valid real numbers: 9.05, 0.06, 78, etc.

Valid use of integers: 9.7**4, V12**-3, V12 ** - 3, etc.

Valid use of non-integer exponents: V23**0.897, etc.

3. Variables

A variable is a named quantity whose value may change during execution of the program. Variables are specified by name.

The name must start with the letter "V" followed by one or two decimal digits representing the number 1 to 40 inclusive. These variables are always considered as real variables.

Example: V1, V23, V39, etc.

In addition the letter "T" is used as a variable. T is the real variable which represents the problem time; T is the time variable of the system solution. "X" is the variable of the differential equation and will be explained later.

4. Expressions

An expression is a string consisting of constants, variables, and function calls interspersed with operators and parentheses. The formulation of an expression is governed by normal mathematical convention and the rules given below. There are two kinds of expressions; arithmetic and logical. The value of an arithmetic expression is a real number. The value of a logical expression is always a truth value, TRUE or FALSE. Expressions may appear in If Statements, Assignment Statements and in Differential Equation Statements.

a. Arithmetic Expressions

(1) The arithmetic operators are as follows:

Arithmetic Operator	Definition
**	Exponentiation
*	Multiply
/	Division
+	Addition
-	Subtraction

(2) Rules for the Construction of Arithmetic Expressions:

(a) All desired computations must be specified explicitly. For example, to multiply the variables V1 and V3 one must write V1*V2 and not V1V2.

(b) No two numeric operators may appear in sequence. (Note: ** is considered as a single operator). Thus A* - V19 is illegal but may be corrected by the addition of parenthesis obtaining A*(-V19).

(c) Parenthesis may be used as desired for clarification and to force the proper order of computations.

(d) A mathematical function, together with its argument, may be used and is treated as a variable.

(3) Order of computation in Arithmetic Expressions:

Operation	Hierarchy
Evaluation of functions	1st
Exponentiation	2nd
Multiplication and division	3rd
Addition and subtraction	4th

(4) Examples of Arithmetic Expressions:

```
(A*V3 + 5.7)/EXP(V1*B*T)
-A*(1.0 - X**2)*X' - B*X
A*X'*ABS(X') + SIGN(X)
```

b. Logical Expressions

(1) Relational Operators. There are six relational operators. They, and the logical operators which follow, must always start and end with a period. The relational operators are binary operators which operate on arithmetic expressions and express an arithmetic condition which can be either true or false. The relational operators are as follows:

Relational Operator	Definition
.GT.	Greater than
.GE.	Greater than or equal to

.LT.	Less than
.LE.	Less than or equal to
.EQ.	Equal to
.NE.	Not equal to

Examples

V1 .LT. X'

0.94 .GE. EXP(A** - 0.5)

X .LT. -A .OR. X .GT. A

(2) Logical Operators. The three logical operators are .NOT. , .AND. , and .OR. . Their use is defined in the ordinary sense of logic.

(3) Rules for Construction of Logical Expressions:

(a) Relational operators may only operate on expressions which, when evaluated, have a numeric value.

(b) Logical operators may only operate on expressions which, when evaluated, have a logical value (i.e., TRUE or FALSE).

(c) Relational operators may not appear in sequence.

(d) Two logical operators may appear in sequence only if the second one is the logical operator .NOT. .

(e) A relational and a logical operator may not appear side by side in an expression.

(f) As in Arithmetic Expressions, parenthesis may be added as desired for clarification and to force the proper order of calculations.

(g) If the logical operator .NOT. operates on an expression with more than one term, the expression must be enclosed with parenthesis.

(4) Order of Computations in Logical Expressions.

Operation	Hierarchy
Evaluation of functions	1st
Exponentiation	2nd
Multiplication and division	3rd
Addition and subtraction	4th
.LT., .LE., .EQ., .NE., .GT., .GE.	5th
.NOT.	6th
.AND.	7th
.OR.	8th

If two operators have the same hierarchy, then they are used in the order they appear left to right.

(5) Examples of Logical Expressions.

```
(V1**2.3 .GE. A) .AND. (X' .LT. 0)

T .GT. 1.5

T .GT. 1.5 .OR. (X .NE. 0.5) .AND. .NOT. (X .LT. X')

(T .GT. 1.5 .OR. (X .NE. 0.5)) .AND. .NOT. (X .LT. X')
```

These last two are not equivalent since the added set of parenthesis has altered the computation order in the last case.

5. Mathematical Functions

The language provides for the use of various mathematical functions. The present library is listed below. One calls these functions in an implicit manner by simply using the proper function name, followed by its argument in parenthesis, as an ordinary variable. The argument itself may be any valid mathematical expression which in turn may contain function calls.

a. Examples of Use.

```
EXP(V1)
EXP(V1*V3/SQRT(SIN(X/X')))
(SIN(X) + COS(X**2))/3.75
```

b. The various available functions and their preassigned names are expressed below.

Name	Description of Function
SIN	Sine of an angle in radians
COS	Cosine of an angle in radians
TAN	Tangent of an angle in radians
ABS	The absolute value of the argument
EXP	Exponential of the argument
LN	The natural logarithm of the argument
LOG	The base 10 logarithm of the argument
INT	Truncates the real number to an integer then reconverts it to a real.
SIGN	Magnitude of 1.0 with the sign of the argument.

6. Arithmetic Assignment Statements

The goal in developing the above language elements was to make use of them as the building blocks for Arithmetic Assignment Statements, Control Statements, and Differential Equation Statements. This section will discuss the Arithmetic Assignment Statement.

The end result of using an Arithmetic Assignment Statement is to compute a new value for a variable. The general form of an Arithmetic Assignment Statement is a = b

where: a is a valid variable name (written without a sign).

b is any mathematical expression following the above guidelines.

= is an order to compute the value of the expression b on the right, and then give that value to the variable named on the left.

A lengthy discussion can be made on the difference between the Arithmetic Assignment Statement and a mathematical statement of

equality. It is perhaps unfortunate that an "=" is traditionally used in both cases. However if one keeps the above definition in mind, there should be no confusion. For someone completely new to computer programming methods, it is recommended that one of the many good fundamental FORTRAN manuals be consulted. The formation of and use of the Arithmetic Assignment Statement is identical with that of FORTRAN.

Examples:

Let $V1 = \frac{-1}{2x} + \frac{a^2}{4x^2}$

then the coded version is:

```
V1 = -1./(2.*X) + A**2/(4*X**2)
```

Let $V5 = (A + Bx^3)^{2/5}$

then the coded version is:

```
V5 = (A + B*X**3)**(2.0/5.0)
```

Let $V9 = (SIN^2 x + COS^2 x)^{\frac{1}{2}}$

then the coded version would be:

```
V9 = (SIN(X)**2 + COS(X)**2)**0.5.
```

Let $V23 = \frac{1}{\cos x} + \log \left| \tan \frac{x}{2} \right|$

then the coded version would be:

```
V23 = 1.0/COS(X) + LOG(ABS(TAN(X/2.0))).
```

Let $V5_{new} = V5_{old} + \left| ax^2 \right|$

then a coded version would be:

```
V5 = V5 + ABS(A*X*X).
```

7. Control Statements

a. The Skip Statement

The control statements are of two general types. One is a simple unconditional transfer statement called a Skip Statement. Its form is:

SKIP N

where: N is an integer that specifies how many statements should be skipped.

b. The Arithmetic IF Statement

The other more powerful control statement is the Arithmetic IF Statement. This is essentially the statement that one is familiar with in FORTRAN. Its form is:

IF(a) THEN b;

where: a is any valid logical expression as previously defined.

b may be an Arithmetic Assignment Statement, a Skip Statement, or a Differential Equation Statement.

It commands the computer to evaluate the logical expression a. If a is true then do b, otherwise skip b and go on to the next statement. The use of the keyword THEN is optional as is the ";".

c. Examples of Valid Control Statements

```
SKIP 3
IF(X  .GT.  X' ) THEN SKIP 4
IF(X  .LE.  ABS(X')) THEN V1 = A*X**2
IF(X  .GE.  0.0) THEN  X'' + 0.2*X' + X = 0.0
```

8. Differential Equation Statements

a. Notation

Differential Equation Statements are intended to be as natural as possible. The apostrophe notation, which is widely used,

has been adopted. Thus this program used x'' as the symbol for the second derivative of x.

Other derivatives of X are handled in an analogous manner. With the exception of this notation for the derivatives, the remainder of the differential equation is written using the construction methods used for mathematical expressions. Note that in the case of a Differential Equation Statement, the statement actually expresses an equation. Thus for this statement only, the equal sign denotes equality rather than an assignment.

Example: X''' + A*X'*ABS(X') + X**2 = V1*T

b. Writing a Differential Equation Statement

The form of a Differential Equation Statement is:

$$a \pm b = c$$

where: a denotes the highest derivative of X written using the notation of the previous section.

b and c denote any valid mathematical expressions using any or all of the constants and variables and using any or all of the lower derivatives of X as variables.

It can be seen that the rules for writing b and c are extremely liberal. Thus the type of differential equation that can be handled is very general. The differential equations may be extremely non-linear, and have coefficients that vary with time and/or position.

c. Examples

The above generality can best be demonstrated by a few examples. Valid constructions are:

X''' + A*X'' + 0.75*X' + 0.0

X'' + (X*X')**X + SIN(X) = 10.0

X'''' + X''' + X''/X = LOG(X'''**0.5)

C. USING THE LANGUAGE

1. General Comments

The goal of all the above elements of the language has been to develop tools for use in defining the differential equations of a system. The above language features make it possible to handle a wide variety of system types. This section demonstrates, by use of several examples, how to use the elements of the language to describe various systems. The series of examples will start with simple applications and then progress into more involved descriptions.

2. A Simple Differential Equation

The simplest, but still very useful and powerful, form of system description consists of one Differential Equation Statement. Let us take for example Van der Pol's

Equation: X'' - A*(1.0 - X**2)*X' + B*X = 0.0

This one data card, or line on the graphics terminal, completely describes the differential equation with the exception of A and B. The variables A and B are specified on an additional data card. When it is desired to alter the values of A and B they are changed independently of the system equations. Considerable execution time is saved by eliminating recompilation of the system equations when it is only desired to alter the value of a coefficient. If, however, one did not anticipate varying coefficients A and B, then the differential equation could be coded as follows, for example:

X'' - 0.95*(1.0 - X**2)*X' + 1.7*X = 0.0

Assume now that it was desired that the first coefficient be 0.90 vice 0.95. There is still no problem, the required recompilation is transparent to the user. The on line user will only notice a somewhat longer delay time before the solution starts.

3. <u>More than One Differential Equation Statement</u>

In most control system applications more than one differential equation is needed to describe a given system. However, only one differential equation is applicable at a given moment in the solution. For example, due to a particular type of non-linearity, the selection of the proper differential equation may be dependent on the sign of the velocity or position. In another situation, the selection of the differential equation may be a function of time. Numerous examples of various control system descriptions appear in Chapter V.

For purposes of illustration, we shall assume the following system:

If X is less than -0.2
then the differential equation is

X'' + 0.2*X' + X = 0.2

If X is greater than 0.2
then the differential equation is

X'' + 0.2*X' + X = -0.2

If X is neither of the above
then the differential equation is

X'' + 0.2*X' + X = 0

The above system could be coded as

```
IF(X .LT. - 0.2) THEN X'' + 0.2*X' + X = 0.2
IF(X .GT. 0.2) THEN X'' + 0.2*X' + X = -0.2
X'' + 0.2*X' + X = 0.0
```

During the execution of the solution, when the differential equation is referred to, the Interpreter will test the argument of each IF statement starting at the top. It will use the differential equation corresponding to the first condition it finds to be true. Note that the third statement is not preceded by a condition. The effect of the third statement is that the condition for its use is always true. Thus if the Interpreter does not use one of the first two differential equations it will use the third.

To further clarify the selection of the right differential equation, consider the following:

```
X'' + 0.2*X' + X = 0.0
IF(X .LT. -0.2) THEN X'' + 0.2*X' + X = 0.2
IF(X .GT. 0.2) THEN X'' + 0.2*X' + X = -0.2
```

This is the same set of statements as before except that the bottom statement has been moved to the top. However, the result is drastically different. The condition for use of the present first differential equation is true by default. Thus it will be used regardless of the value of X. Again, the point to be stressed is, only one differential equation can be used at a given moment; the one used will be the first one with a true condition.

4. Using the Remaining Features of the Language

Often it is convenient to be able to define and calculate some variables before the Differential Equation Statement. Consider the following coded system description:

```
IF((4.0*X + X') .GE. 0.0) THEN X'' + 0.02*X' + 0.3 = 0.0
IF((4.0*X + X') .LT. 0.0) THEN X'' + 0.02*X' - 0.3 = 0.0
```

Using features of the language previously described this could be recoded as follows:

```
V1 = + 0.3;  IF((4.0*X + X')  .LT.  0.0)  THEN V1 = -V1
X'' + 0.02*X' + V1 = 0.0
```

The semicolon allows more than one statement per line. This recoded version has the same effect as the previous version. Note that there is only one differential equation. The V1 used in the differential equation is defined in the line above in such a way that the effect is the same as the two differential equations of the first version.

For this system an even simpler coding is possible. X'' + 0.02*X' + 0.3*SIGN(4.0*X + X') = 0.0. This example demonstrates the versatility of the language by showing the same system coded in three different ways.

IV. PRINCIPLES AND METHODS OF USE

A. GENERAL COMMENTS

The previous chapters have concentrated on various specific aspects of the program. In Chapter II the internal structure of the program was described. Chapter III described the language associated with use of the program. This chapter will endeavor to pull these together with specific instructions on the use of the program. In Section B those areas common to both the Batch Version and the Interactive Graphics Version will be discussed. Sections C and D will deal with the peculiarities of the two versions respectively.

B. GENERAL PRINCIPLES

1. System Description

One of the first tasks for the user is to encode the system equations in the form usable by the computer. One of the primary features of the program is that this is done in a very natural way. For simple systems this will involve writing only a single equation. For systems with a complicated control strategy the user may have to write what amounts to a small program to completely specify it. This general area was covered with examples in Section C of Chapter III. Chapter V will further amplify this with a large number of complete examples.

2. Use of the Constants

The use of the constants was described in Chapter III. However for the completeness of this Chapter they will be mentioned again. Often when preparing to study a particular system it may be

desired to vary a parameter over several values. One way to accomplish this is to vary the numerical value in the system equations. However this amounts to starting a whole new problem and the system equations must be recompiled.

An easier way to accomplish this parameter variation is to use the constants. These are the constants labeled A - H. When writing the system equation use the constants in place of an actual numerical value and define them separately as desired. When it is desired to change the parameter, redefine the associated constants.

Another handy use of these constants is when the same number is to be used several places in the system description. In this use a constant would be defined once and then used as often as desired.

3. <u>The Phase Plane Window</u>

The output of the program is a phase plane. The slopes and solutions are presented on this phase plane. This phase plane can be visualized as a window in which the user has control over the placement, size, and scales to be used in its construction. The controls are independent in the horizontal and vertical directions. Following are the control parameters and their use:

X-SCALE	The horizontal scale in units per inch.
Y-SCALE	The vertical scale in units per inch.
X-CENTER	The X coordinate of the center of the window.
Y-CENTER	The Y coordinate of the center of the window.
X-SIZE	The horizontal dimension in inches.
Y-SIZE	The vertical dimension in inches.

4. Solution Parameters

The remaining values to be specified are the parameters of the solution. These include the initial time, time step, final time, and the initial conditions.

In most cases, the user will let the initial time be zero and the final time be some positive value with an associated positive time step. However, this is not required. In fact the final time may be less than the initial time. In this later case the time step would be negative.

The program can handle systems up to eighth order. Thus there are provisions for specifying up to eight initial conditions. This limitation to eighth order was arbitrarily set by the ease of handling the initial values on input. Eight initial values fit nicely on one input card.

C. SPECIFIC DETAILS ON THE BATCH VERSION

1. Planning for the Batch Run

For the batch version the user must plan the entire run beforehand. One may be uncertain of some parameters of the problem on the first run. If this is the case, the user should take advantage of the flexibility of this program on the first submission. That is, the first submission should include combinations of values such that when the results are returned one will know what values to choose for the next run. For example, take the problem of the choice of time step and final time. Here it would be good to make a guess at a suitable time step and final time. Then run an additional two solutions with these times multiplied by ten and divided by

ten. The added effort for these additional solutions would be trivial. When the results come back it will then be quite easy to decide on the best values.

The same idea can apply to other areas such as scaling. The time required to punch the added cards for a few extra runs can save considerable time lost to turn-around-time if an added exploratory run is required.

2. Data Card Arrangement

Cards 1-2	Plot title including your name (6A8) Format
3	X-SCALE(Units/inch), X-CENTER (Coordinate of X-CENTER) X-SIZE (Inches) Y-SCALE (As above) Y-CENTER (As above) Y-SIZE (As above) These six items are placed on one card in the above order using a (6F10. 0) Format. If (X-SIZE .GT. 9.0) then the plot is rotated CCW on the paper by a quarter turn. Due to paper size only one of the dimensions may exceed 9. 0 inches and must not be more than 15. 0 inches.
4	SIZE of slope markers (inches, 0. 2 works nicely), NUMBER per inch (3 recommended) In (F10. 4, I1) Format
5, 6, 7, 8...	EQUATIONS TO BE USED: Following the rules of Chapter III (80A1 Format).

Then follow four cards per solution desired as described below:

-First-	VALUES of A, B, C, D, E, F, G, H (8F10. 4) Format
-Second-	Integration info (INITIAL TIME, TIME STEP, and FINAL TIME) (900 Solution steps Max), (3F10. 4) Format.
-Third-	INITIAL CONDITIONS for X, X', X'', . . . in (8F10. 4) Format

-Fourth- A "WHATS NEXT CARD" use the following code:

0 - There are no requests following.
1 - The following are the second and third cards with new integration and initial condition information only. I want this new solution to be plotted on the present graph.
2 - The following is another four cards with data for another solution of the present system on a new graph.
3 - The following card starts a completely new problem starting with the plot title cards.

3. Sample Data Deck

Following is a typical sample deck of input cards. The comments enclosed within parenthesis are not part of the cards but are included in this listing for clarification only. These cards accomplish the following.

The system X'' + A*X' + SIN(X) = 0 is compiled and two solutions are obtained with A = 0.3. Then A is changed to 0.7 and two more solutions are run using the same time information and initial conditions as with A = 0.3. Then a completely new problem (a saturated servo system) is defined.

The input cards used are:

```
NELSON, H. G.      Pendulum     (GRAPH TITLE - 1st LINE)
X'' = A*X' = SIN(X) = 0         (GRAPH TITLE - 2nd LINE)
2.0      0.0      8.0      2.0      0.0      6.0 (WINDOW INFO)
0.2      4                      (SLOPE MARKER SIZE AND NO.)
X'' + A*X' + SIN(X) = 0         (SYSTEM EQUATION)
                                (END OF SYSTEM EQUATIONS)
0.3                             (VALUE OF A)
0.0      0.05     25.0          (TIME INFO)
-7.0     3.0                    (INITIAL CONDITIONS)
1                               (WHAT'S NEXT SIGNAL)
0.0      0.05     25.0          (TIME INFO)
-7.0     5.0                    (INITIAL CONDITIONS)
2                               (WHAT'S NEXT SIGNAL)
0.7                             (VALUE OF A)
0.0      0.05     25.0          (TIME INFO)
```

```
-7.0      3.0                           (INITIAL CONDITIONS)
1                                       (WHAT'S NEXT SIGNAL)
0.0       0.05      25.0                (TIME INFO)
-7.0      5.0                           (INITIAL CONDITIONS)
3                                       (WHAT's NEXT SIGNAL)
NELSON, H. G.  0902                     (GRAPH TITLE - 1st LINE)
  SATURATED SERVO SYSTEM                (GRAPH TITLE - 2nd LINE)
0.3       0.0       6.0       0.2       0.0       8.0  (WINDOW INFO)
0.2       4                             (SLOPE MARKER SIZE AND NO.)
IF(X.LT.(1.0-2.0)*0.2 THEN X'' + 0.2*X' + X = 0.2
IF(X.GT.0.2) THEN X'' + 0.2*X' + X = -0.2  (SYSTEM EQUATIONS)
X'' + 0.2*X' + X = 0.0
                                        (END OF SYSTEM EQUATIONS)
0.0       0.05      30.0                (TIME INFO)
0.7                                     (INITIAL CONDITIONS)
                                        (WHAT'S NEXT = BLANK = 0, THUS
                                         END OF DATA)
```

D. SPECIFIC DETAILS ON THE INTERACTIVE GRAPHICS VERSION

1. Planning for the Interactive Session

Since the Interactive Version is very easy to use and, as the name implies, is interactive the required preparation is minimal. This statement assumes the first-time user has read this manual. If a mistake is made or a bad guess is made for a parameter value, it can normally be corrected before the run starts. If one notices the error after the run has started, there is still no problem. The program has a large amount of protection built into it. Hopefully, no mistakes or errors by the user can cause the interaction to terminate. Regardless of the immediate results of a mistake, control will return to the user for desired corrections and/or modifications.

2. Check-off List for Loading the Program

Following is a step by step list to be used to load and run the program.

a. Turn on the XDS 9300 Computer (if not already on) using the instructions attached to the console. Also turn on Sense Switch 2.

b. Mount the binary tape of the program on one of the tape drives and advance it to the load point. Set the "unit select" to 0, the "density select" to 556 and the mode switch to automatic. The "unit power", "unit ready", "file protect", and "load point" lights should be on.

c. Place the small card deck as specified in Appendix B in the card reader and press the "power on" and "start" buttons in that order.

d. Ready the printer. On the XDS 9300, verify that "Sense Switch 2" is on. Then press the following buttons:

HALT
RESET
CLEAR and CLEAR FLAGS simultaneously
RUN
CARDS.

e. The card reader should begin reading cards and then the Tape Drive will begin operating. It will take about three minutes to load and link the program.

f. Following the instructions at the AGT's, turn on the AGT unit to be used and load "Gated".

g. Verify that "Gated" is ready by pressing the "Text Edit" button on the moveable button panel. The following should appear at the bottom of the screen:

"TEXT BLOCK SELECT MODE   Block 1"

h. When the XDS console typewriter types "ENTER IDEV = 1 or 2" do the following:

1. Ensure the AGT is ready
2. Note the number of the AGT
3. Type "IDEV = 1*", return
   or "IDEV = 2*", return as applicable.

i. This completes the preparation phase. Although this list appears long or complicated it is quite easy once it has been done a few times. Quite often the user will find both the XDS and the AGT already turned on and ready to go. It is strongly recommended that the initial use of this program be made during working hours when advice and/or help is available.

3. Interacting with the Program

The interaction starts with a question to the user. "Do you desire to start a completely new problem?, answer 'YES' or 'NO' ". Answer "YES" or "NO" on the keyboard and then hit the return key. If you answer "NO", a problem already in the computer will be displayed. This is useful for becoming familiar with the program. It is suggested for the first time user that the answer should be "NO" to that question. Then the new user should try various manipulations on the given problem until a working knowledge of the program is obtained.

After each run the control returns again to this same question. After the first pass the answer to the question gives the user the option of making modifications to the present system already on the screen or starting a complete new problem. If there are only a few changes to be made it is best to selectively edit the present problem. If it is desired to start a new problem or make many changes, it will probably be faster to use the "INPUT" mode. When using the "INPUT" mode, the system prompts the user for the desired information.

The easiest way to learn the system is to use it. Thus the following section is a step-by-step procedure for using the program on several specific problems. It is suggested that the user closely

follow this guide using the given problems. Once the user is familiar with the procedures for using the program and the graphics terminal then he can proceed on his own.

4. Suggested First Session Example

a. Load the program as described in Section 2. The question "DO YOU DESIRE TO START A COMPLETELY NEW PROBLEM?, ANSWER YES OR NO" will appear on the screen.

b. Type "NO", return. At this time the internally stored program will appear on the screen. In addition will be the message "MAKE DESIRED CHANGES AND CHOOSE NEW INITIAL CONDITIONS (TEXT OR LIGHT PEN) TO INITIATE NEW RUN". The signal to the program that the user is ready to proceed is to edit the first line of the initial conditions or to select initial conditions graphically with light pen.

c. If satisfied with the initial conditions already on the screen, make the following null edit. Press the following buttons in succession:

TEXT EDIT
NEXT BLOCK - Check to see that the second line characters are slanted. If not use the NEXT BLOCK and/or PREVIOUS BLOCK until line two is slanted. The slanting indicates that line two is ready to be edited.
GO EDIT - Corrections can now be made as desired to line two. For this time a null correction will be made, i.e., no correction.
TERMINATE EDIT

The program now has all the information it needs to make a run and the above edit of line two has indicated that the program is ready to be run. When the run has terminated the question of step a. will be repeated. A complete loop in the program has now been completed.

d. Again answer by typing "NO", return.

e. Before this next run the goal will be to change the initial conditions to -3.8 and 3.0 respectively. To do this "select" and "go edit" on line two as in step c above. This time a correction will be made. While holding the typewriter key marked "CTRL", space over with the "F" (forward) key until the underscore is immediately to the right of the first initial condition to be corrected. Now release the "CTRL" key. Press the "RUB OUT" key the desired number of times to eliminate the portion to be changed. Then type in the new first initial condition. Be sure to replace as many characters as were removed. This keeps the whole line in the proper alignment. Next, space over and correct the second initial condition using the same keys as used to correct the first initial condition. When you are satisfied with the line press the terminate edit button or the "RETURN" key on the typewriter. Either one will terminate the edit and initiate the next run. The end of this run will bring you back to the question of step a.

f. Again answer by typing "NO", return.

g. The goal of this run is to select a set of initial conditions using the light pen. Press in sequence the "graphics edit", "go edit", and "cursor on" buttons. Insure that the light pen is turned on by checking for the pilot light on the box at the end of the light pen cable. Take the light pen, point it at the cursor, press the little white button on the light pen and lead the cursor around the screen. When you are familiar with positioning the cursor, select with the cursor a desired initial condition, then press the "move" and then "draw" buttons. With the light pen move the cursor to another initial condition on the screen. There should be a dot remaining at the location of the first initial condition pair selected. Pressing the "move", "draw" buttons

will select the present cursor position as a second initial condition pair. This can be repeated up to fifteen times if desired. When you are through selecting initial conditions, terminate the edit. This will, as when selecting the initial conditions numerically, initiate the run. A solution will now be obtained for each of the selected initial conditions and then control will return again to the question of step a.

h. Again answer "NO".

i. This time change A to 3.0 . To do this press "text edit" and then press next block until the characters in the line containing the A tilt. You are now ready to edit this line using the method of step e above.

j. When through editing A initiate the next run using the technique of steps c, e, or g above.

k. Once more answer "NO".

l. This time it is desired to edit the system equation to obtain X'' + X*X' + X = 0. Press the "text edit" and then the "next block" button about eight times until the equation characters slant indicating they have been selected. Make the modification as before. You do not need to replace as many characters as you removed when editing the equations. When ready initiate the next run using one of the previously discussed methods.

m. When the solution has been completed and the question of step a returns answer "YES". In response to each question make responses as suggested below. After each response hit the return key.

SYSTEM EQUATIONS:

```
V1 = 4.0*X + X'
IF(V1.GE.0.0)    THEN X'' + 0.02*X' + 0.3 = 0.0
X'' + 0.02*X' - 0.3 = 0.0
blank line

X-SCALE        =  0.2
X-CENTER       =  0.0
X-SIZE         =  6.0
Y-SCALE        =  0.2
Y-CENTER       =  0.0
Y-SIZE         =  8.0
A - H             hit return, will default to zero
INITIAL TIME   =  0.0
TIME STEP      =  0.04
FINAL TIME     =  10.0
FIRST I. C.    =  0.6
remainder of I. C.'s, hit return,
```

n. Now the program will give the question "MAKE DESIRED CHANGES (TEXT OR LIGHT PEN) TO INITIATE NEW RUN". At this point you can proceed as in steps c. through l. above as desired. Thus if any corrections need to be made as a result of incorrect responses to the above questions they can be made at this time. When ready to initiate the run use the method of step c. It is best to select the initial conditions numerically for the first run after changing the window parameters. Initial conditions selected with the light pen take on the corresponding values of that point relative to the current picture on the screen.

o. You have at this time exercised all the control functions needed for full interactive control of the program.

V. APPLICATIONS TO VARIOUS PROBLEMS

A. GENERAL COMMENTS

The purpose of this chapter is to demonstrate the use of the program by using it to study several classes of control system problems. It is not the intention of this chapter to derive the applicable equations, but rather to show the use of the program in their solution. If the purpose of the system is to reduce the error to zero or a minimum, then it is convenient to use error as the dependent variable instead of the regular output variable.

In the following examples the problem will be defined first then the data cards will be shown, followed by the resultant phase plane. Although these examples will have been produced using the hard plots of the Batch Version, the input data is identical with that of the Interactive Graphics Version.

B. VARIOUS SINGLE EQUATION EXAMPLES

Example 1

Van der Pol's Equation

```
X'' - A*(1.0 - X**2)*X' + B*X = 0.0
```

WHERE:

A = 1.0
B = 1.0

PLOT PARAMETERS:

XSCALE	=	1.0	YSCALE	=	1.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
NELSON, H. G.
X'' - A*(1.0 - X**2)*X' + B*X = 0.0
1.0       0.0       6.0       1.0       0.0       8.0
0.2       4
 X'' - A*(1.0 - X**2)*X' + B*X = 0.0

1.0       1.0
0.0       0.1       20.0
0.01      0.0
1
0.0       0.1       20.0
1.0       0.0
1
0.0       0.1       20.0
2.5       3.0
1
0.0       0.1       20.0
2.0       -3.3
1
0.0       0.1       20.0
-2.5      3.0
1
0.0       0.1       20.0
-2.5      -2.0
```

Example 1: $X'' - A*(1.0 - X**2)*X' + B*X = 0.0$

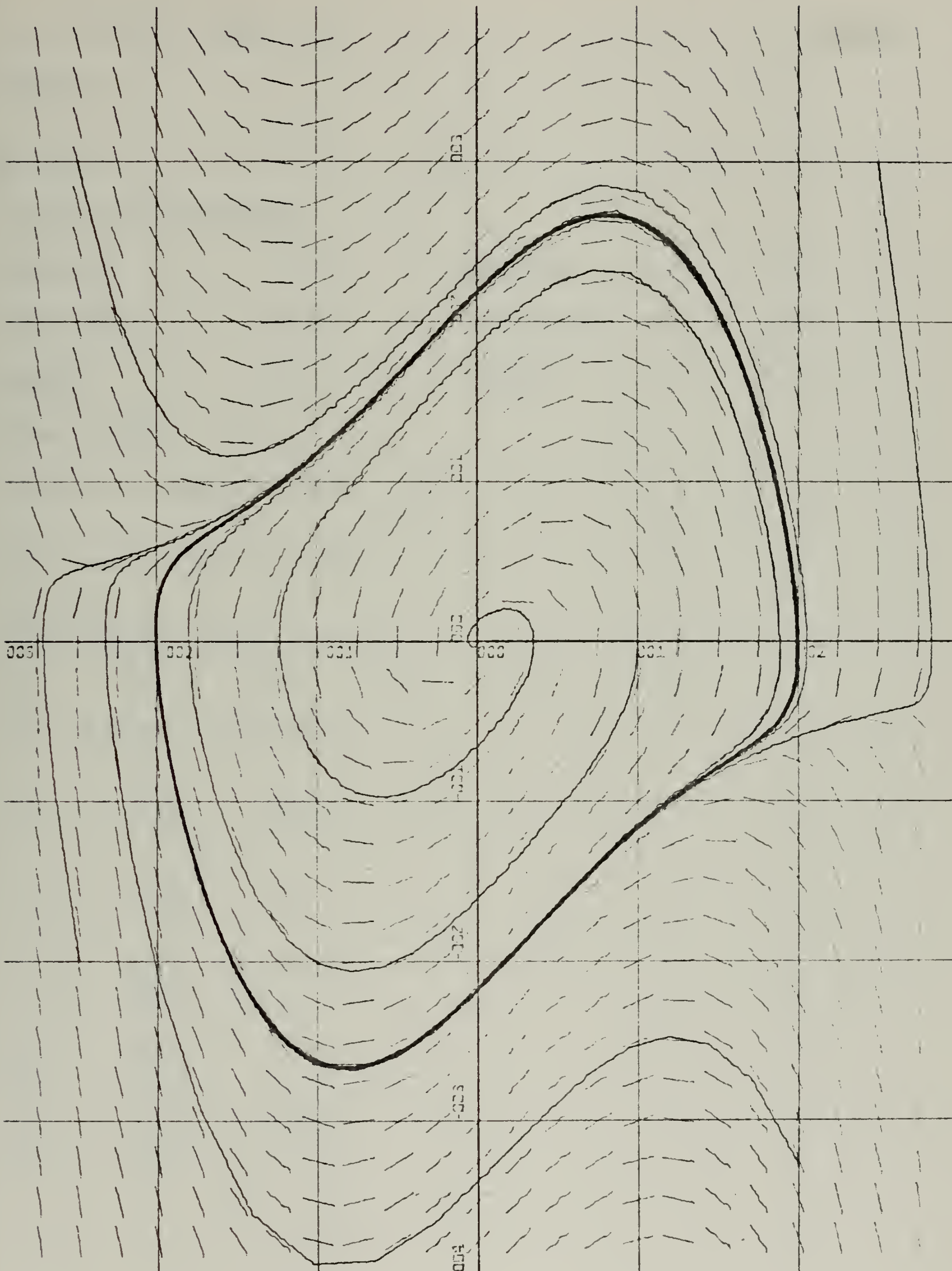

Fig. 5-1 X-SCALE = 1.0 units/inch
Y-SCALE = 1.0 units/inch

Example 2:

X'' + A*X*X' + B*X = 0. 0

WHERE:

A = 1. 0
B = 1. 0

PLOT PARAMETERS:

XSCALE	=	1. 0	YSCALE	=	1. 0
XCENTER	=	0. 0	YCENTER	=	0. 0
XSIZE	=	6. 0	YSIZE	=	8. 0

Size of slopes = 0. 2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
NELSON, H. G.
X'' + A*X*X' + B*X  =  0. 0
1. 0       0. 0       6. 0       1. 0       0. 0       8. 0
0. 2       4
X'' + A*X*X' + B*X = 0. 0

1. 0       1. 0
0. 0       0. 02      15. 0
           1. 0
1
0. 0       0. 02      15. 0
           2. 0
1
0. 0       0. 02      15. 0
           3. 0
1
0. 0       0. 02      15. 0
2. 5       -2. 0
1
0. 0       0. 02      15. 0
2. 5       -3. 5
```

Example 2: X'' + A*X*X' + B*X = 0. 0
A = 1. 0, B = 1. 0

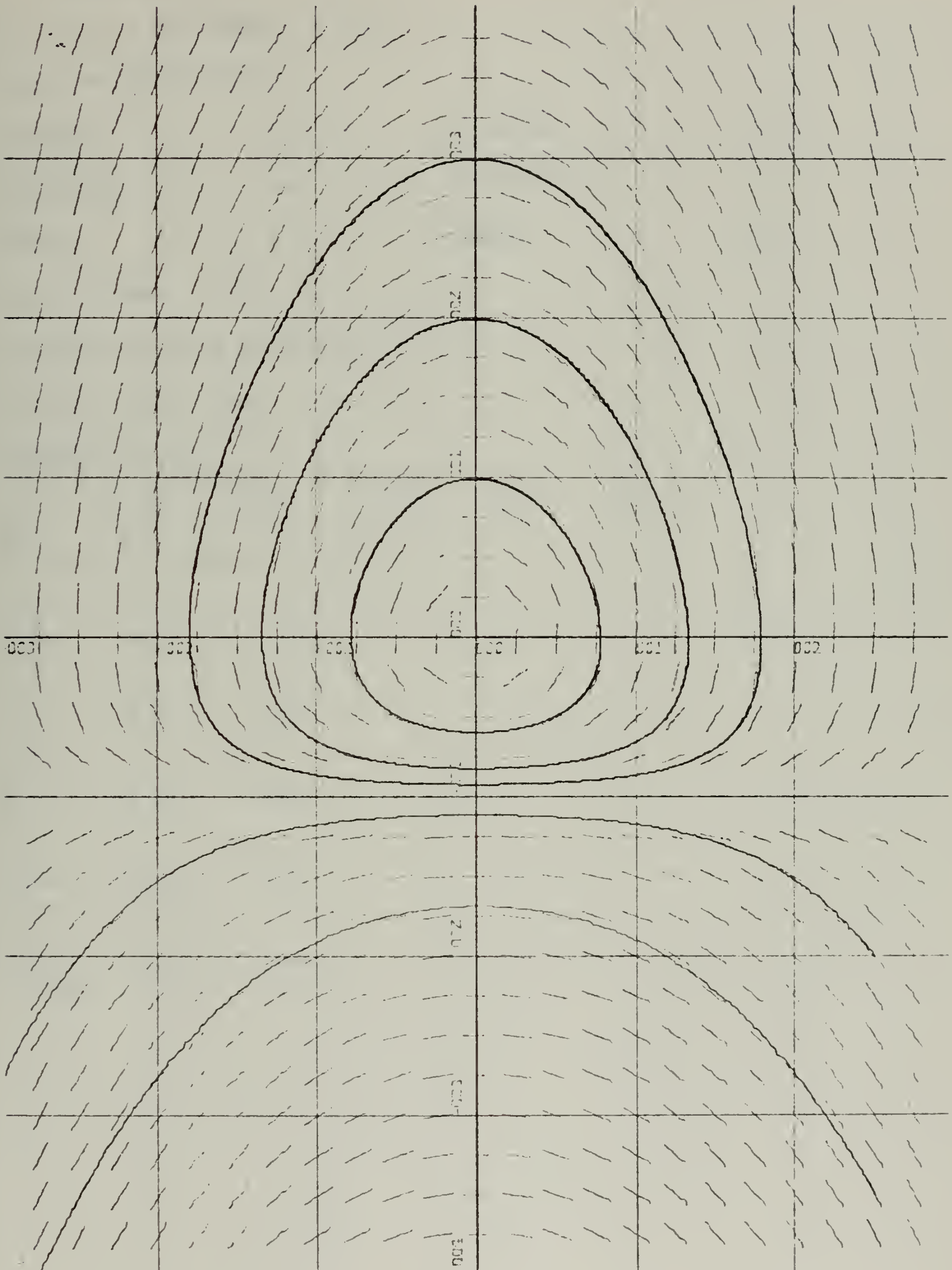

Fig. 5-2 X-SCALE = 1. 0 units/inch
Y-SCALE = 1. 0 units/inch

Example 3:

X'' + (1.0 - X**2)*X' + X = 0.0

PLOT PARAMETERS:

XSCALE	=	1.0	YSCALE	=	1.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

Size of slopes = 0.2

Number of slopes per inch = 4

The input cards used:

```
NELSON, H. G.
X'' + (1.0 - X**2)*X' + X = 0.0
1.0       0.0       6.0       1.0       0.0       8.0
0.2       4
X'' + (1.0 - X**2)*X' + X = 0.0

0.0
0.0       0.01      8.0
1.0
1
0.0       0.01      10.0
1.5
1
0.0       0.01      8.0
2.0
1
0.0       0.01      8.0
2.25
1
0.0       0.01      8.0
-2.05
```

Example 3: X'' + (1.0 - X**2)*X' + X = 0.0

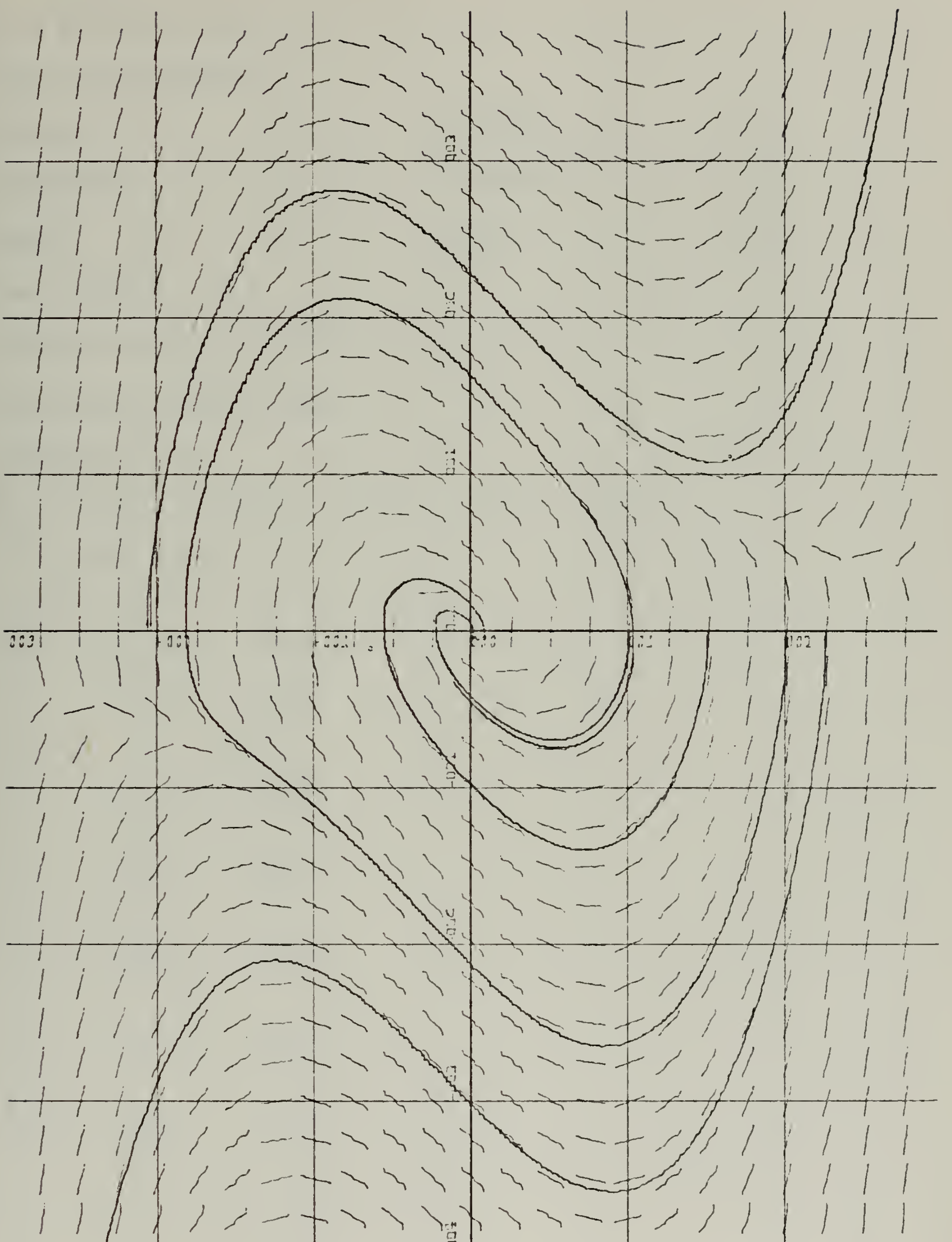

Fig. 5-3 X-SCALE = 1.0 units/inch
Y-SCALE = 1.0 units/inch

Example 4:

X'' + X'**2 + X = 0.0

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	1.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	8.0	YSIZE	=	6.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
NELSON, H. G.
X'' + X'**2 + X = 0.0
2.0       0.0       8.0       1.0       0.0       6.0
0.2       4
X'' + X'**2 + X = 0.0

0.0
0.0       0.05      40.0
-4.0      2.5
1
0.0       0.05      40.0
-7.0
1
0.0       0.05      40.0
1.0       3.0
1
0.0       0.05      40.0
-5.0
1
0.0       0.05      40.0
-3.0
1
0.0       0.05      40.0
-1.0
1
0.0       0.05      40.0
0.4       2.5
```

Example 4: X'' + X'**2 + X = 0. 0

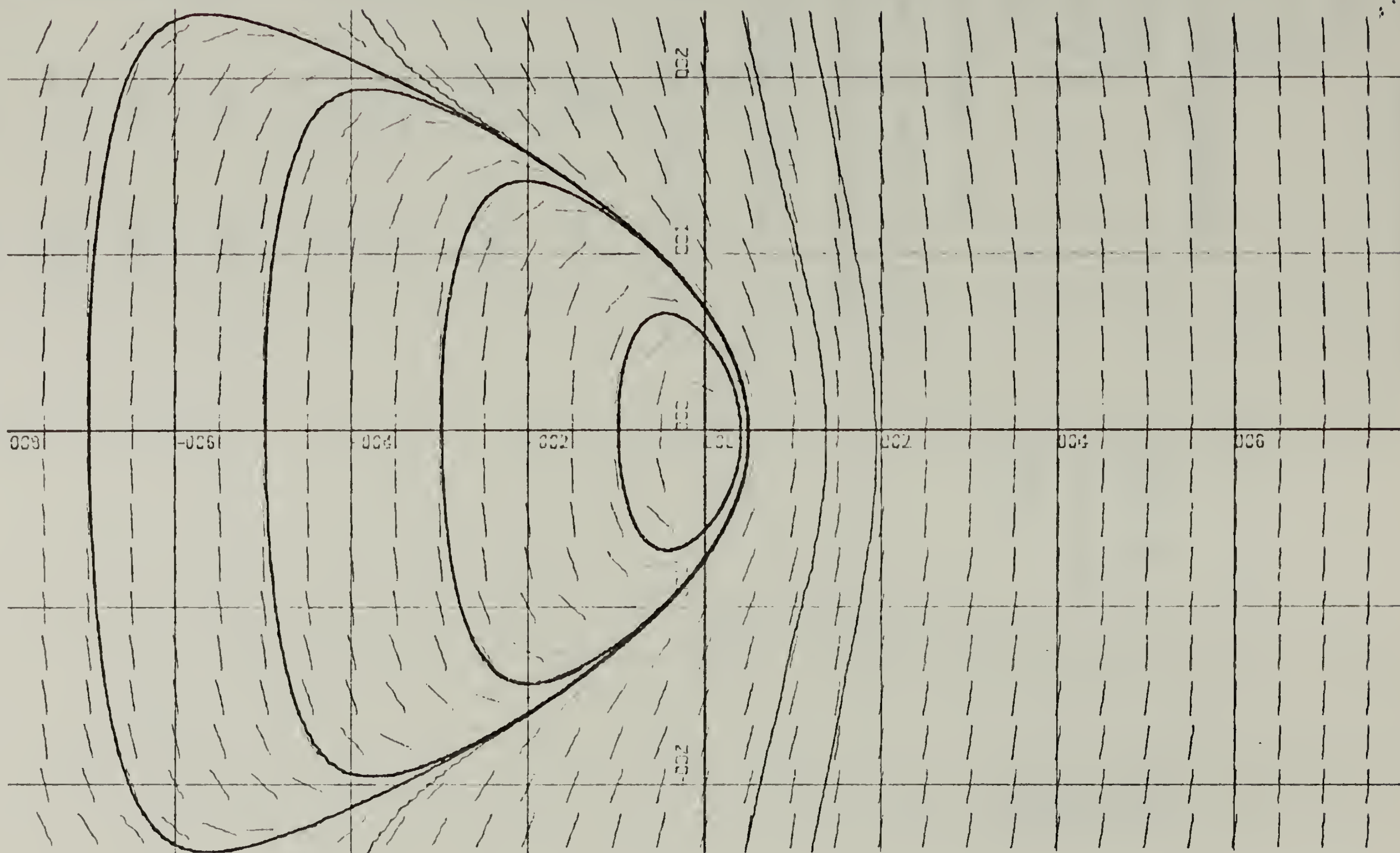

Fig. 5-4 X-SCALE = 2. 0 units/inch

Y-SCALE = 1. 0 units/inch

Example 5:

X'' + A*X + B*X**3 = 0.0

WHERE:

A = -1.0
B = 0.25

PLOT PARAMETERS:

XSCALE	=	1.0	YSCALE	=	1.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	8.0	YSIZE	=	6.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
 NELSON, H. G.
 X'' + A*X + B*X**3 = 0.0
1.0     0.0     8.0     1.0     0.0     6.0
0.2     4
 X'' + A*X + B*X**3 = 0.0

-1.0    0.25
0.0     0.02    18.0
1.0     1.2
1
0.0     0.02    18.0
1.0     2.0
1
0.0     0.04    30.0
0.0     0.03
1
0.0     0.02    18.0
0.3
1
0.0     0.02    18.0
-0.3
1
0.0     0.02    18.0
1.0
1
0.0     0.02    18.0
-1.0
1
0.0     0.02    18.0
-1.5
1
0.0     0.02    18.0
1.5
```

Example 5: X'' + A*X + B*X**3 = 0.0

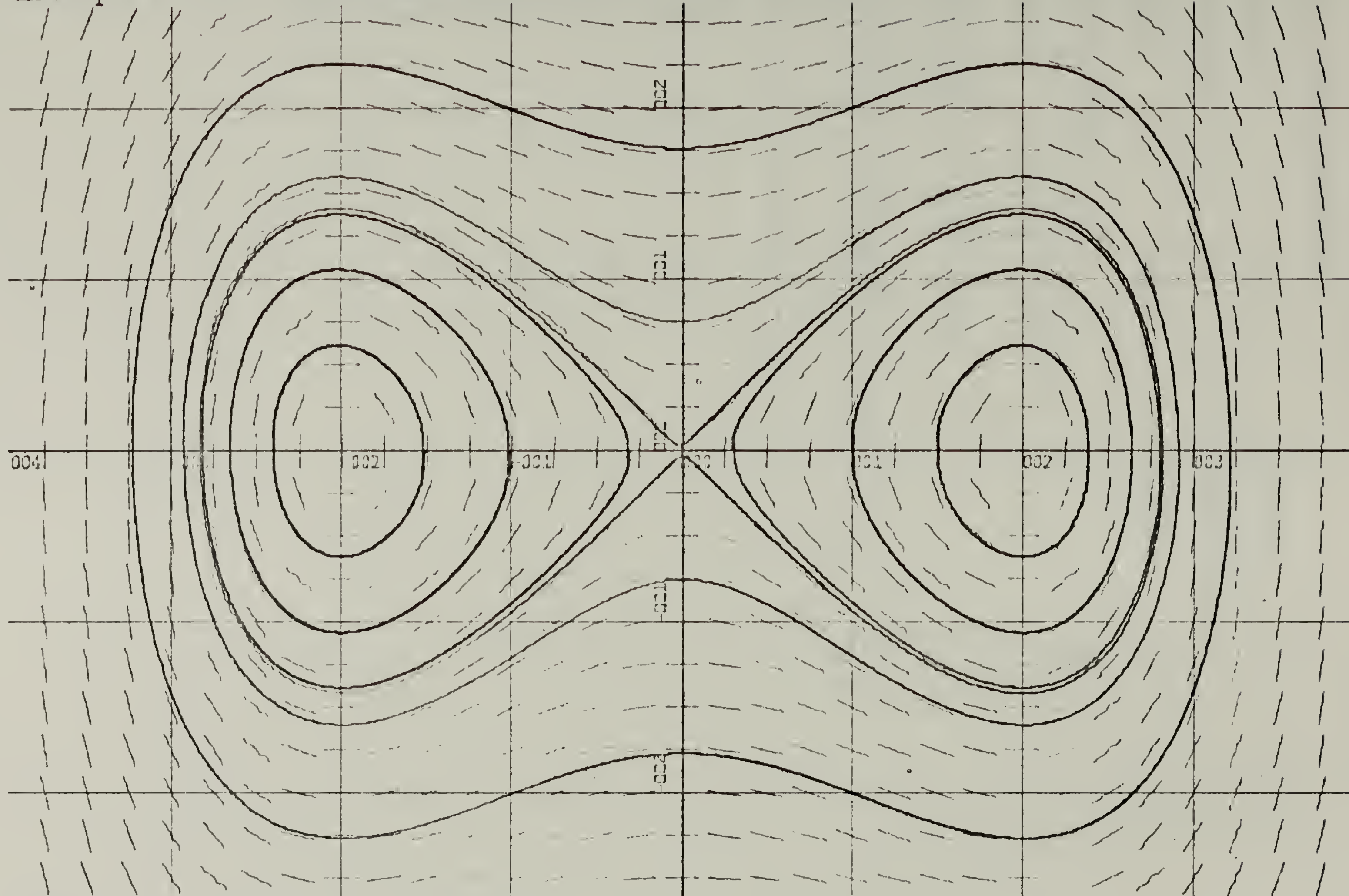

Fig. 5-5 X-SCALE = 1.0 units/inch

Y-SCALE = 1.0 units/inch

Example 6:

X'' + (1.0 - ABS(X))*X' + X = 0.0

PLOT PARAMETERS:

XSCALE	=	1.0	YSCALE	=	1.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
 NELSON, H. G.
 X'' + (1.0 - ABS(X))*X' + X = 0.0
1.0      0.0      6.0      1.0      0.0      8.0
0.2      4
 X'' + (1.0 - ABS(X))*X' + X = 0.0

0.0
0.0      0.02     15.0
1.0
1
0.0      0.02     15.0
1.5
1
0.0      0.02     15.0
2.0
1
0.0      0.02     15.0
2.8
1
0.0      0.02     15.0
-2.5
```

Example 6: X'' + (1.0 - ABS(X))*X' + X = 0.0

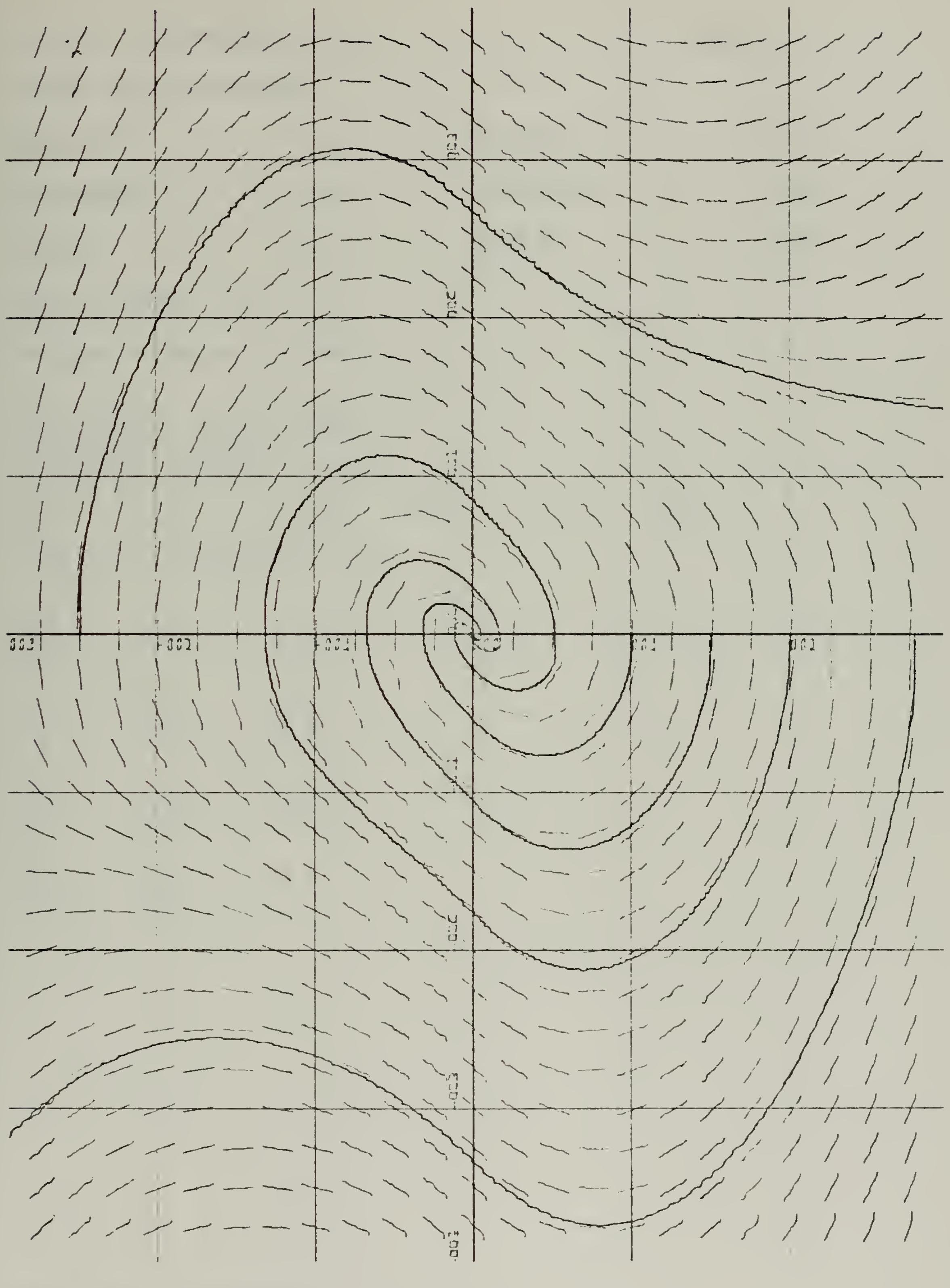

Fig. 5-6   X-SCALE = 1.0 units/inch
Y-SCALE = 1.0 units/inch

Example 7:

X'' + X' + X*ABS(X) = 0.0

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	2.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
 NELSON, H. G.
 X'' + X' + X*ABS(X) = 0.0
2.0     0.0     6.0     2.0     0.0     8.0
0.2     4
 X'' + X' + X*ABS(X) = 0.0

0.0
0.0     0.01    8.0
3.8
1
0.0     0.01    8.0
-3.0
1
0.0     0.01    8.0
5.0
1
0.0     0.01    8.0
-5.5
```

Example 7: X'' + X' + X*ABS(X) = 0.0

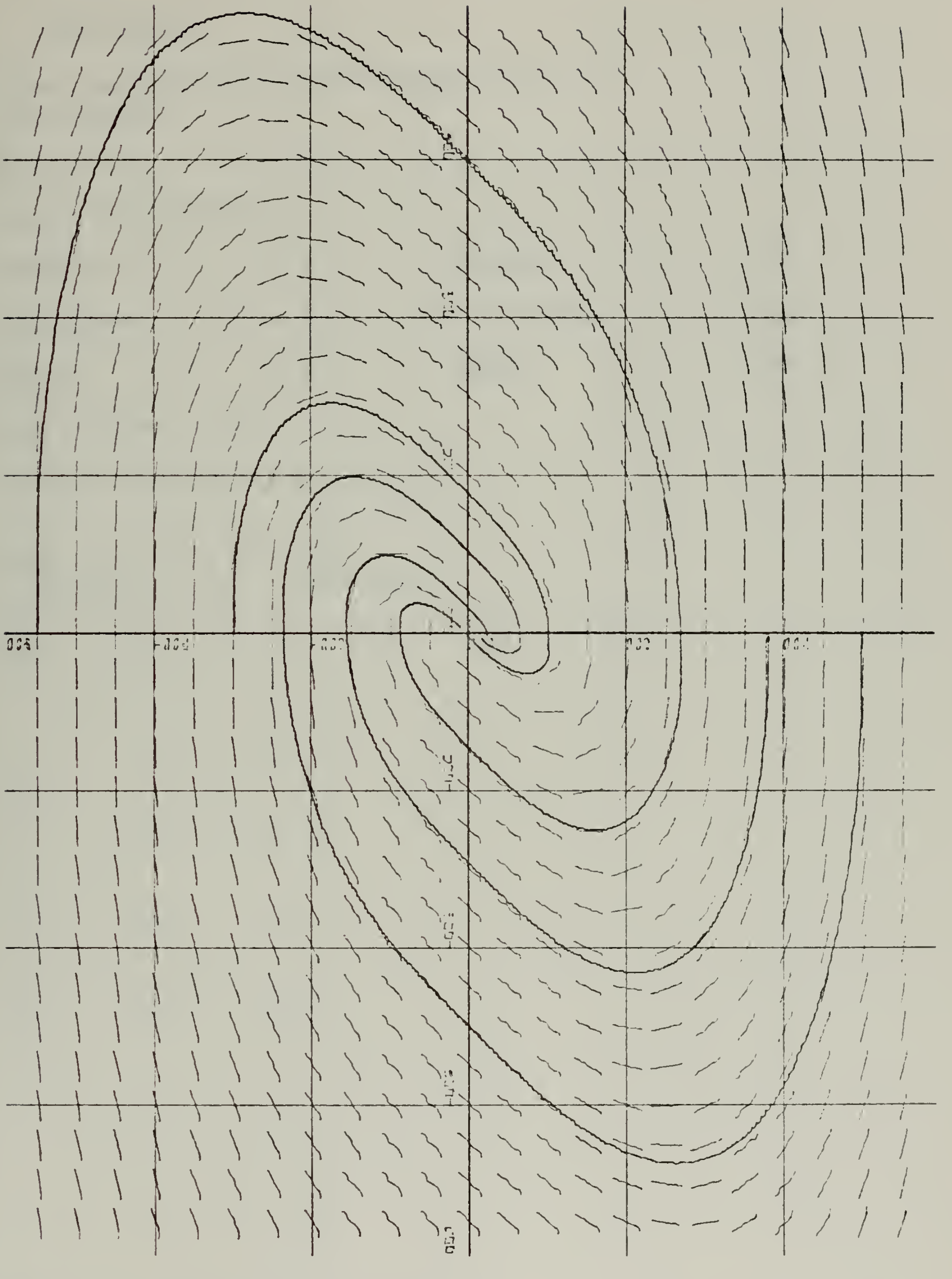

Fig. 5-7 X-SCALE = 2.0 units/inch
Y-SCALE = 2.0 units/inch

C. VARIOUS TYPES OF PENDULUM EXAMPLES

Example 8:

Linear approximation to the pendulum, g/l = 1
No Damping

X'' + X = 0

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	2.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
 NELSON, H. G.   PENDULUM
   X'' + X = 0
2.0      0.0      6.0      2.0      0.0      8.0
0.2      4
   X'' + X = 0


0.0      0.02     10.0
2.0
1
0.0      0.02     10.0
1.0
1
0.0      0.02     10.0
4.0
1
0.0      0.02     10.0
5.5
```

Example 8: $X'' + X = 0$

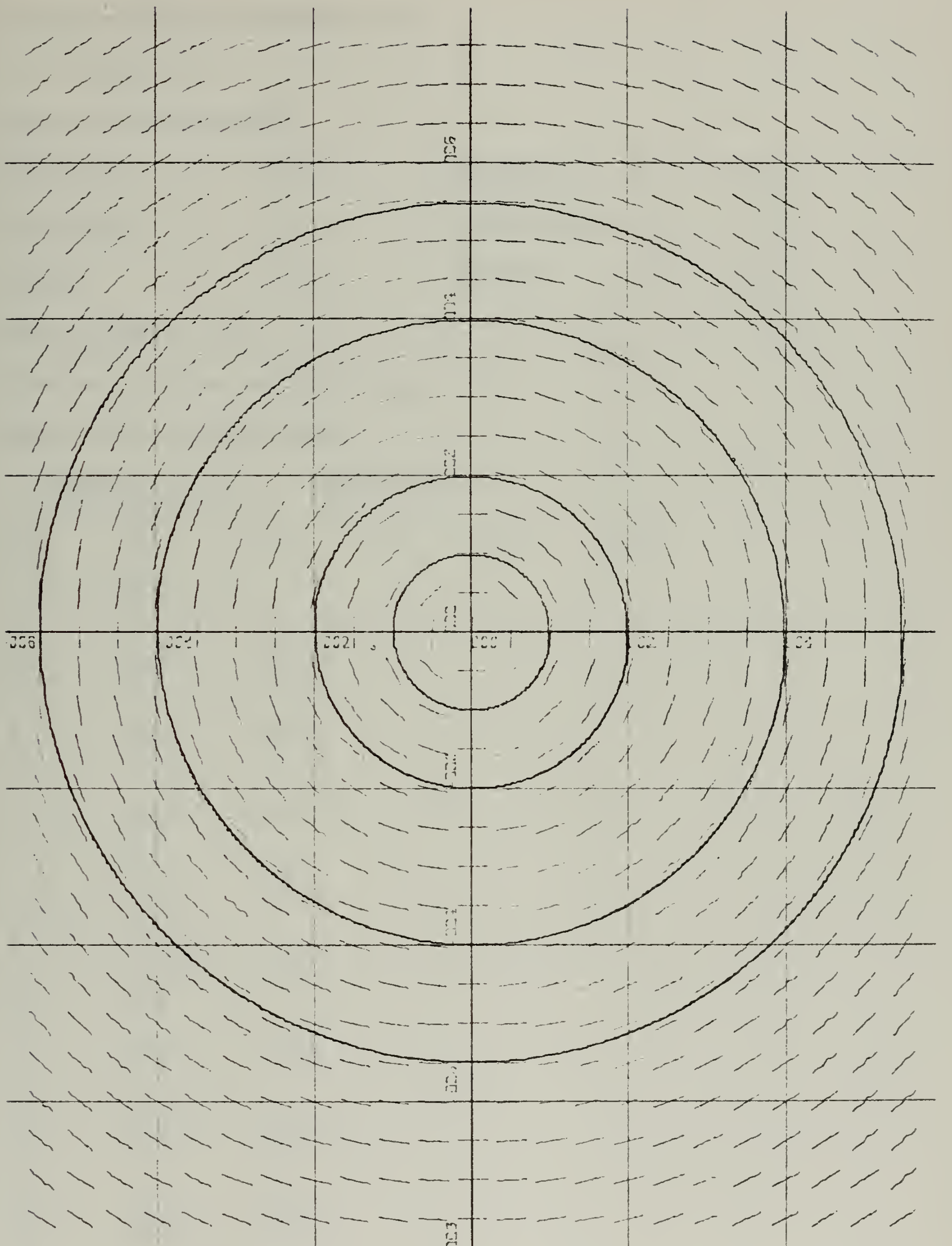

Fig. 5-8 X-SCALE = 2.0 units/inch

Y-SCALE = 2.0 units/inch

Example 9:

Pendulum without damping, g/1 = 1

X'' + SIN(X) = 0

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	2.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	8.0	YSIZE	=	6.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
  NELSON, H. G.     PENDULUM
  X'' + SIN(X) = 0
2.0      0.0      8.0      2.0      0.0      6.0
0.2      4
  X'' + SIN(X) = 0

0.0      0.02     10.0
-6.0     1.0
1
0.0      0.02     15.0
-6.0     2.0
1
0.0      0.02     10.0
-6.0     3.0
1
0.0      0.02     10.0
-6.0     4.0
1
0.0      0.02     10.0
0.0      1.0
1
0.0      0.02     10.0
6.0      1.0
1
0.0      0.02     15.0
6.0      -2.0
1
0.0      0.02     10.0
6.0      -3.0
1
0.0      0.02     10.0
6.0      -4.0
```

Example 9: $X'' + SIN(X) = 0$

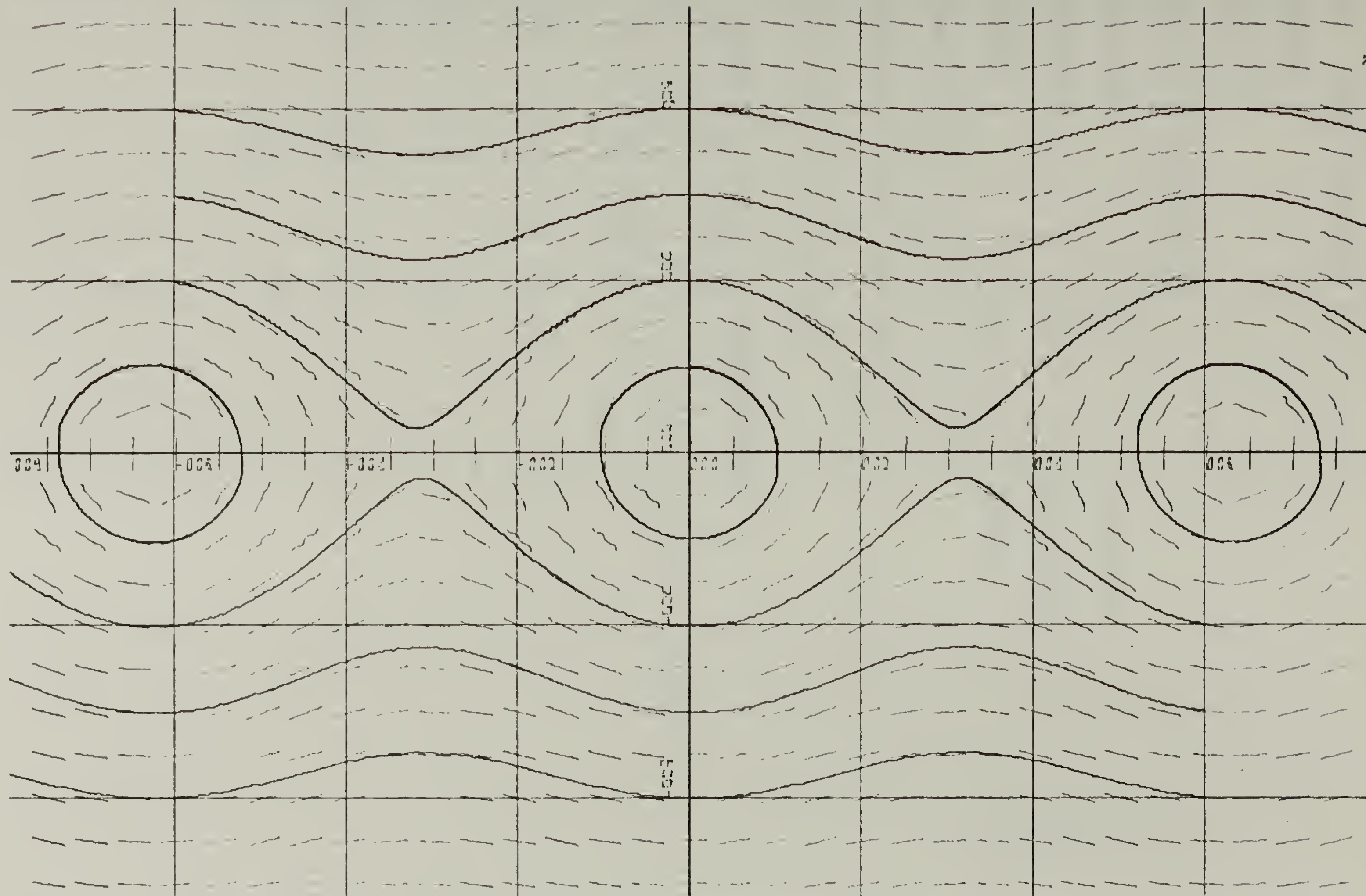

Fig. 5-9 X-SCALE = 2.0 units/inch

Y-SCALE = 2.0 units/inch

Example 10:

Pendulum with velocity damping.

X'' + A*X'+ SIN(X) = 0

WHERE:

A = 0.3

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	2.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	8.0	YSIZE	=	6.0

Size of slopes = 0.1

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
  NELSON, H. G.     PENDULUM
  X'' + A*X' + SIN(X) = 0
2.0      0.0      8.0      2.0      0.0      6.0
0.1      4
  X'' + A*X' + SIN(X) = 0

0.3
0.0      0.05     25.0
-7.0     3.0
1
0.0      0.05     25.0
-7.0     5.0
1
0.0      0.05     25.0
-7.0     2.0
1
0.0      0.05     25.0
6.0      -5.0
```

Example 10: X'' + A*X' + SIN(X) = 0

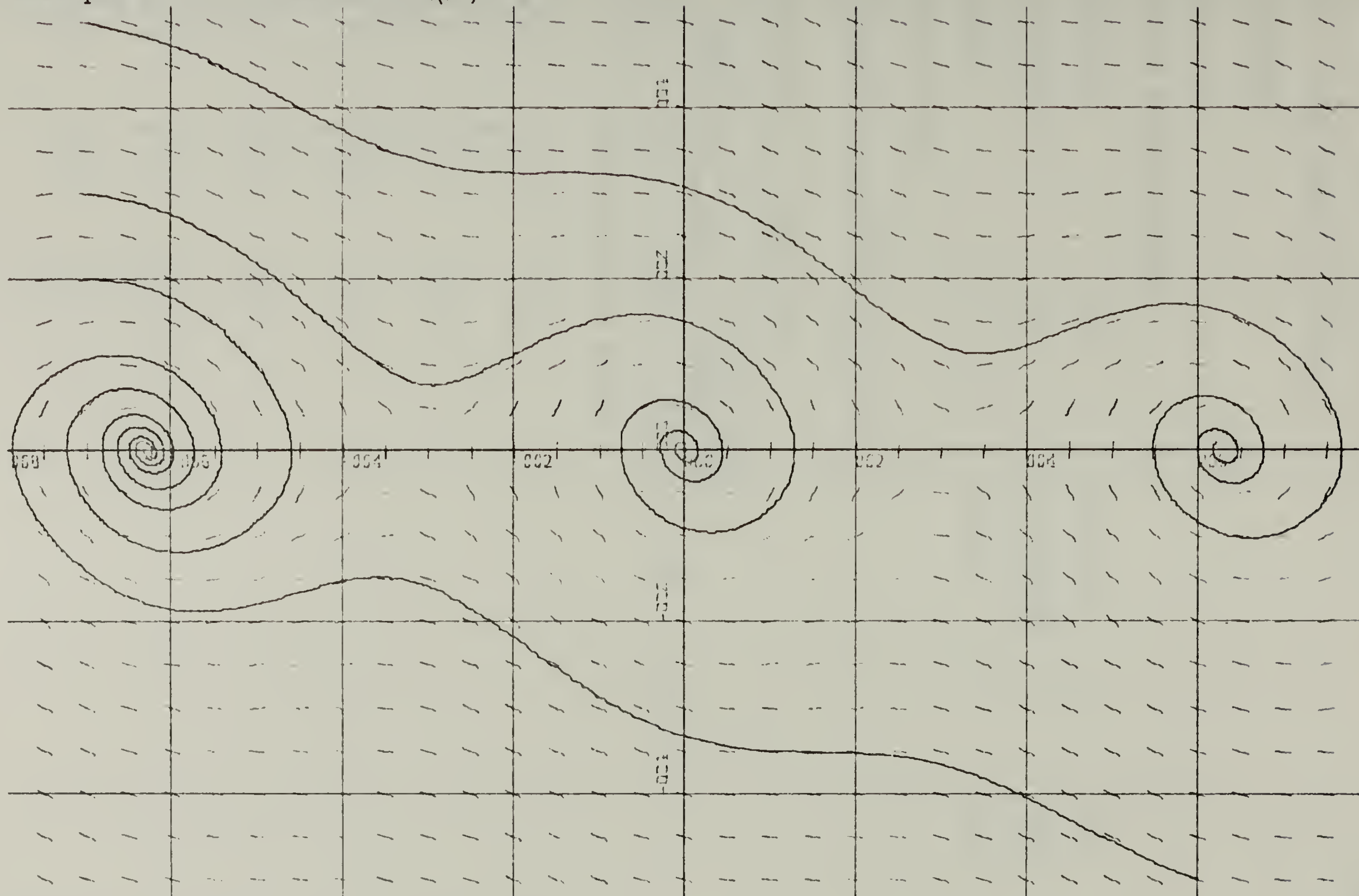

Fig. 5-10 X-SCALE = 2.0 units/inch

Y-SCALE = 2.0 units/inch

Example 11:

Pendulum with velocity squared damping.

X'' + A*X'*ABS(X') + SIN(X) = 0

WHERE:

A = 1.0

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	2.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	8.0	YSIZE	=	6.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
  NELSON, H. G.     PENDULUM
  X'' + A*X'*ABS(X') + SIN(X) = 0
2.0       0.0       8.0        2.0       0.0       6.0
0.2       4
  X'' + A*X'*ABS(X') + SIN(X) = 0

1.0
0.0       0.03      20.0
1.0       4.0
1
0.0       0.03      20.0
-4.0      4.0
1
0.0       0.03      20.0
-7.0      4.0
1
0.0       0.03      20.0
5.0       -4.0
1
0.0       0.03      20.0
2.0       4.0
1
0.0       0.03      20.0
-2.5      -4.0
```

Example 11: X'' + A*X'*ABS(X') + SIN(X) = 0

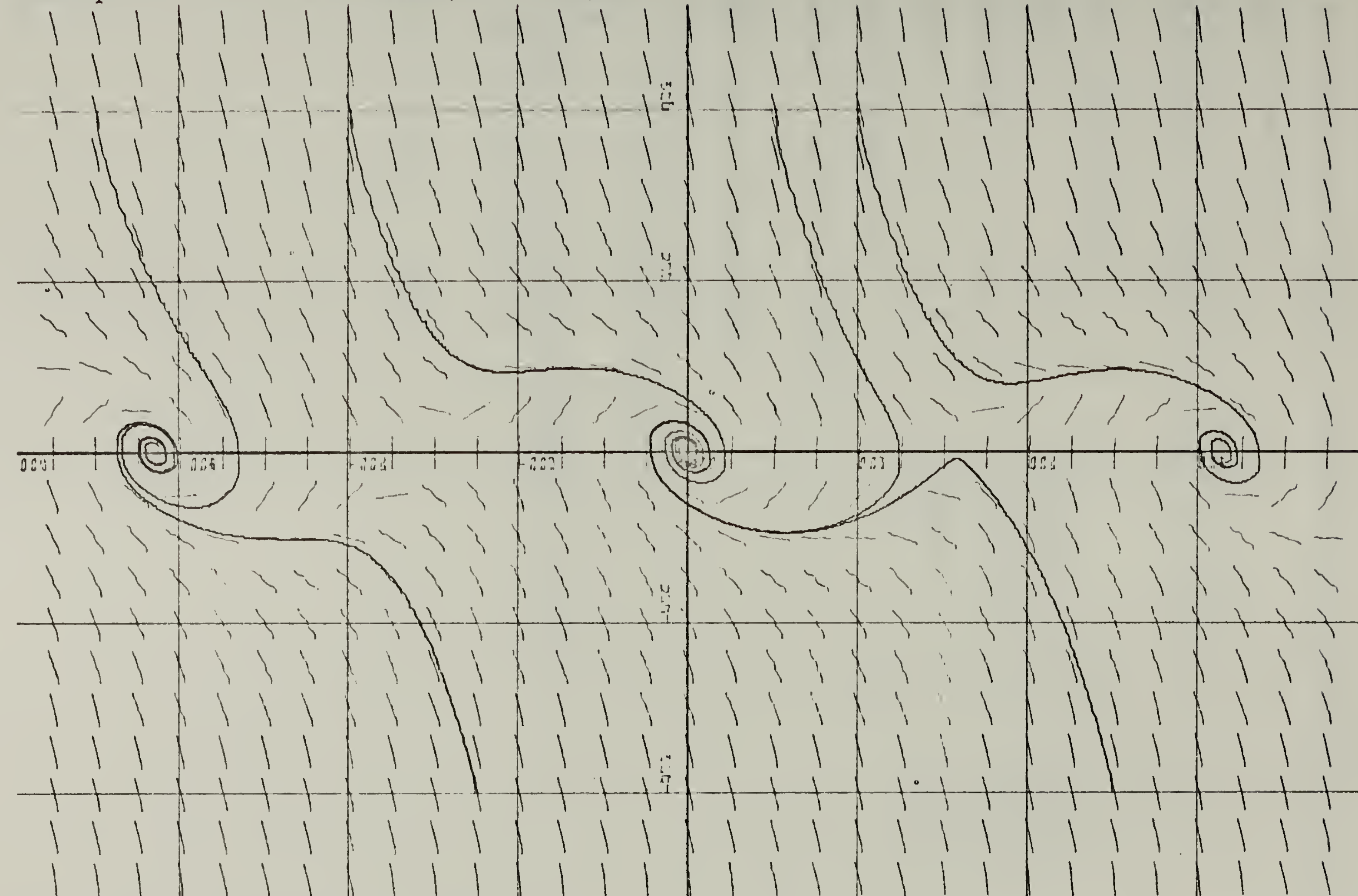

Fig. 5-11 X-SCALE = 2.0 units/inch

Y-SCALE = 2.0 units/inch

D. SATURATED SYSTEMS

Example 12:

A saturated servo system. Saturation occurs when absolute error signal exceeds 0.2.

```
IF(X.LT.-0.2) THEN X'' + 0.2*X' + X = 0.2
IF(X.GT.0.2)  THEN X'' + 0.2*X' + X = -0.2
X'' + 0.2*X' + X = 0.0
```

PLOT PARAMETERS:

XSCALE	=	0.3	YSCALE	=	0.2
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
  NELSON, H. G.  0902
     SATURATED SERVO SYSTEM
0.3      0.0      6.0      0.2      0.0      8.0
0.2      4
  IF(X.LT.(1.0-2.0)*0.2) THEN X'' + 0.2*X' + X = 0.2
  IF(X.GT.0.2) THEN X'' + 0.2*X' + X = -0.2
  X'' + 0.2*X' + X = 0.0


0.0      0.05      30.0
0.7
```

Example 12: Saturated Servo System

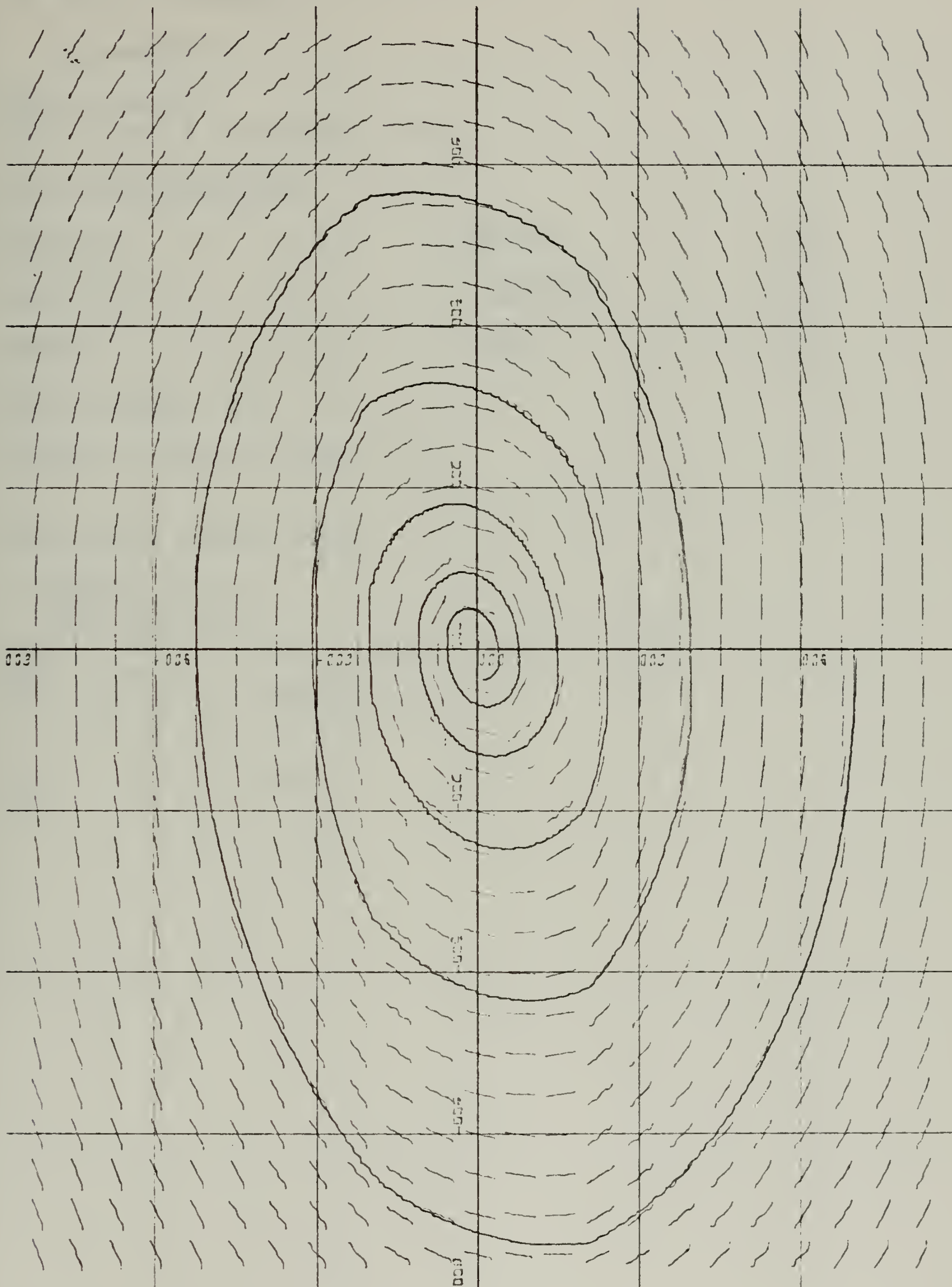

Fig. 5-12 X-SCALE = 0.3 units/inch

Y-SCALE = 0.2 units/inch

E. RELAY SYSTEMS

Example 13:

IDEAL RELAY
X'' + 0. 3*X' + 1. 5*SIGN(X) = 0. 0

PLOT PARAMETERS:

XSCALE	=	1. 0	YSCALE	=	0. 6
XCENTER	=	0. 0	YCENTER	=	0. 0
XSIZE	=	6. 0	YSIZE	=	8. 0

Size of slopes = 0. 2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
   NELSON, H. G.
    IDEAL RELAY
1. 0      0. 0      6. 0      0. 6      0. 0      8. 0
0. 2      4
X'' + 0. 3*X' + 1. 5*SIGN(X) = 0. 0


0. 0      0. 1      85. 0
2. 8
```

Example 13: X'' + 0.3*X' + 1.5*SIGN(X) = 0.0

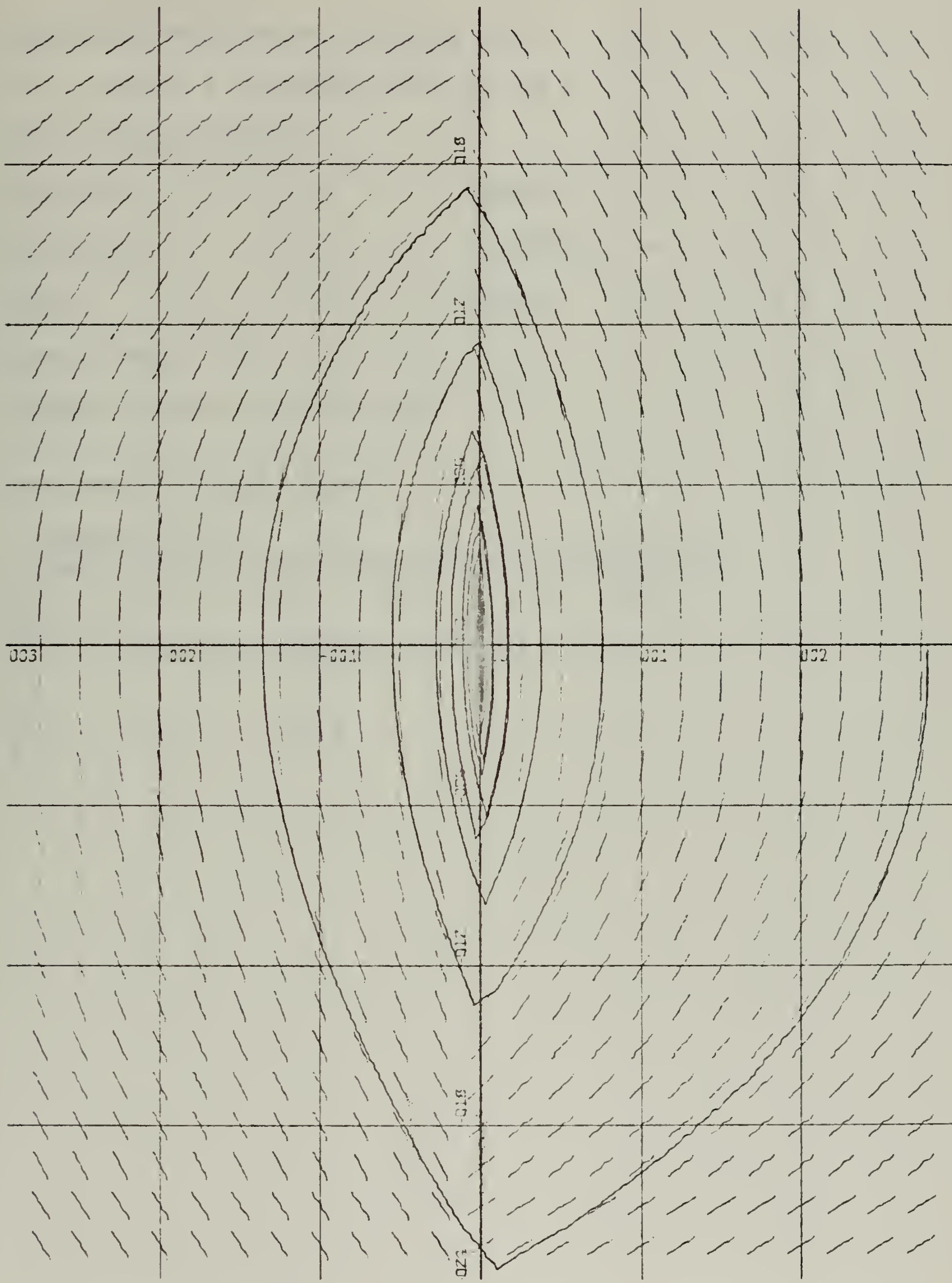

Fig. 5-13 X-SCALE = 1.0 units/inch

Y-SCALE = 0.6 units/inch

Example 14:

Ideal Relay with Rotated Switching Line

X'' + 0.2*X' + 1.0*SIGN(4.0*X - X') = 0.0

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	1.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
  NELSON, H. G.
  IDEAL RELAY WITH ROTATED SWITCHING LINE
2.0     0.0     6.0     1.0     0.0     8.0
0.2     4
  X'' + 0.2*X' + 1.0*SIGN(4.0*X - X') = 0.0

0.0
0.0     0.1          80.0
5.5
```

Example 14: X'' + 0.2*X' + 1.0*SIGN(4.0*X - X') = 0.0

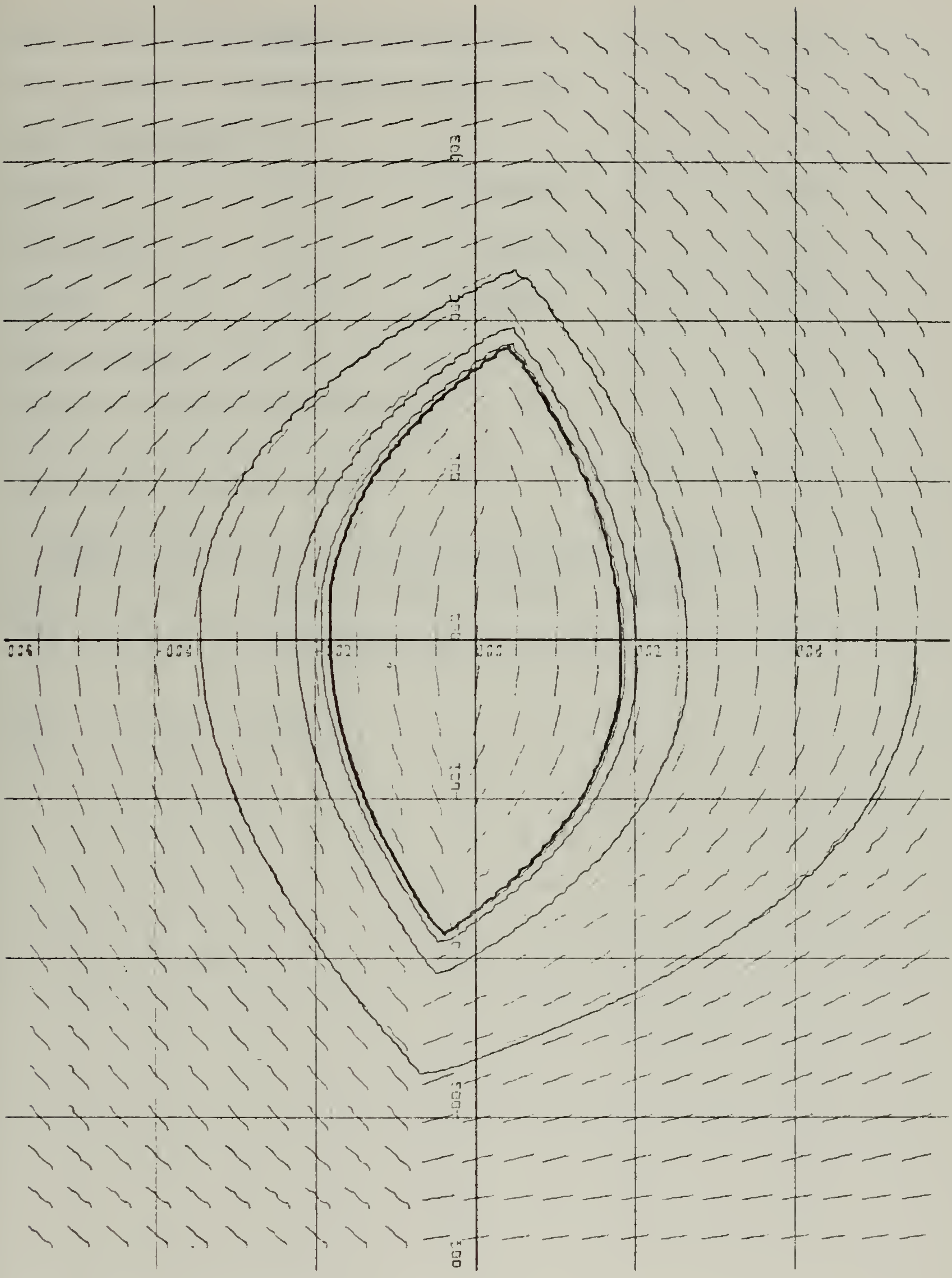

Fig. 5-14 X-SCALE = 2.0 units/inch

Y-SCALE = 1.0 units/inch

Example 15:

Ideal Relay with Rotated Switching Line

X'' + 0.2*X' + 1.0*SIGN(4.0*X + X') = 0.0

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	0.8
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
   NELSON, H. G.
   IDEAL RELAY WITH ROTATED SWITCHING LINE
2.0       0.0       6.0       0.8       0.0       8.0
0.2       4
   X'' + 0.2*X' + 1.0*SIGN(4.0*X + X') = 0.0

0.0
0.0       0.1       20.0
5.5
```

Example 15: X'' + 0.2*X' + 1.0*SIGN(4.0*X + X') = 0.0

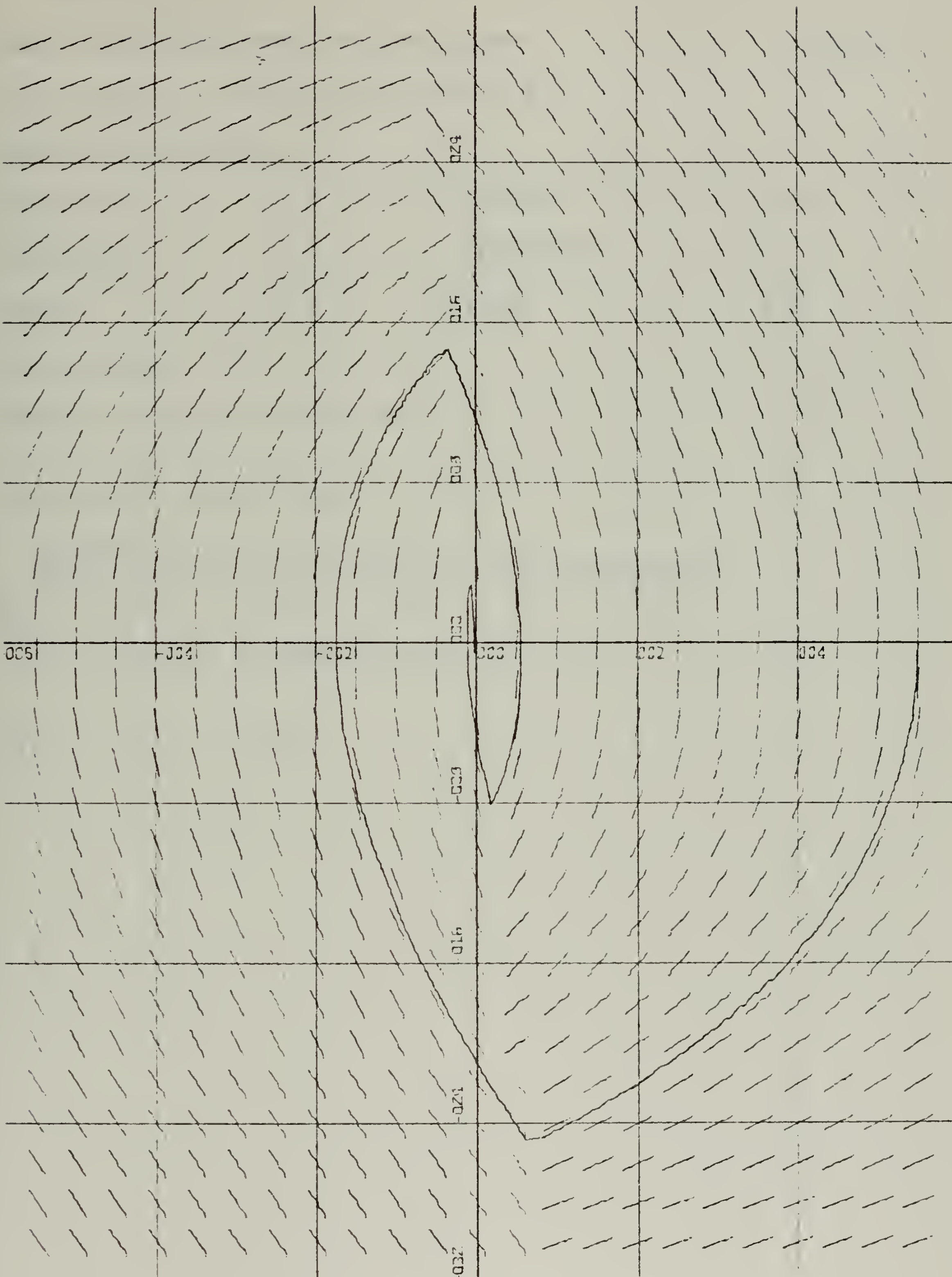

Fig. 5-15 X-SCALE = 2.0 units/inch

Y-SCALE = 0.8 units/inch

Example 16:

Ideal Relay with Rotated Switching Line

X'' + 0.2*X' + 1.0*SIGN(2.0*X + X') = 0.0

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	1.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
   NELSON, H. G.
   IDEAL RELAY WITH ROTATED SWITCHING LINE
2.0      0.0      6.0       1.0      0.0      8.0
0.2      4
   X'' + 0.2*X' + 1.0*SIGN(2.0*X + X') = 0.0

0.0
0.0      0.1      20.0
5.5
```

Example 16: X'' + 0.2*X' + 1.0*SIGN(2.0*X + X') = 0.0

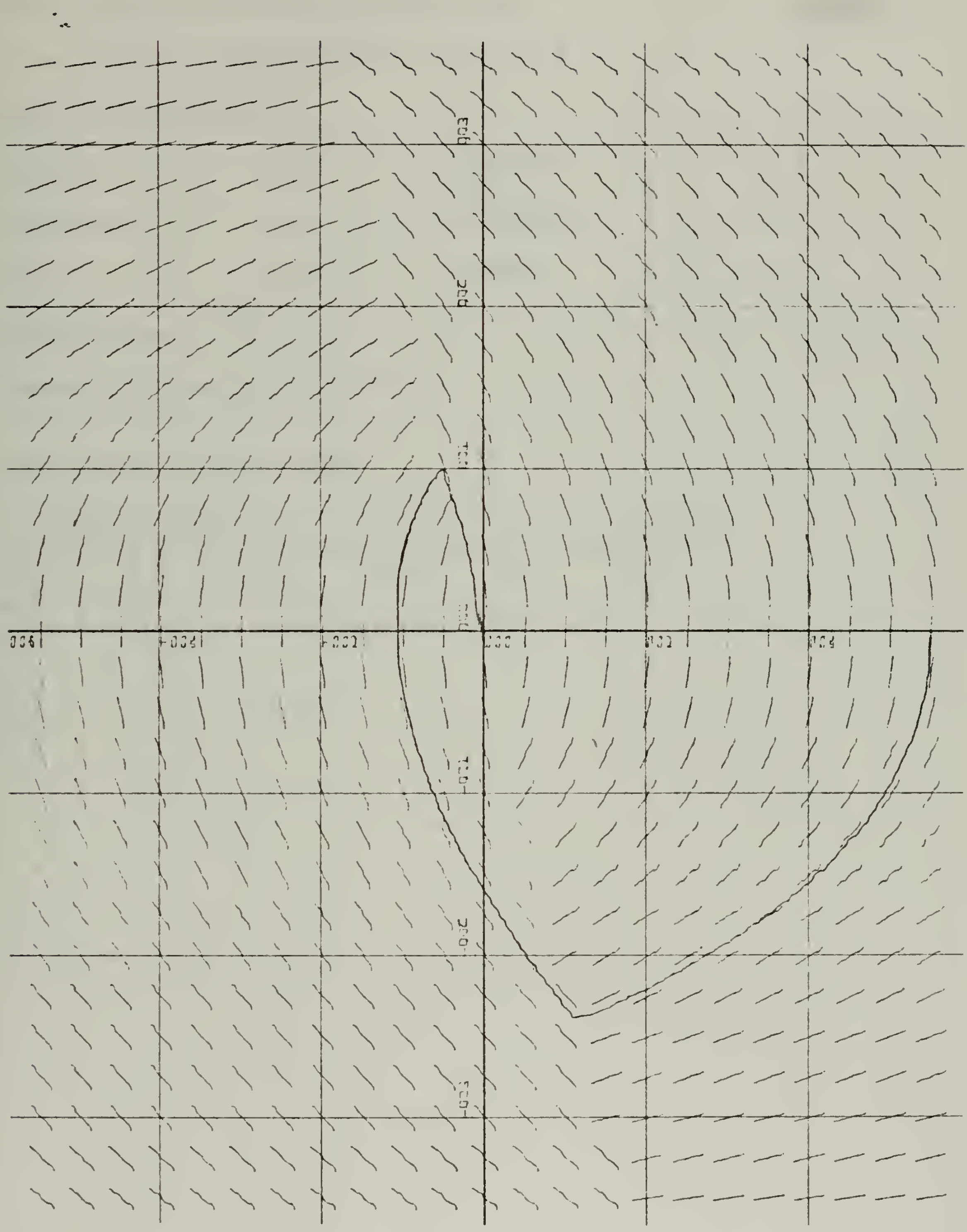

Fig. 5-16 X-SCALE = 2.0 units/inch

Y-SCALE = 1.0 units/inch

Example 17:

Ideal Relay with Rotated Switching Line

X'' + 0.2*X' + 1.0*SIGN(2.0*X - X') = 0.0

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	0.8
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
   NELSON, H. G.
   IDEAL RELAY WITH ROTATED SWITCHING LINE
2.0       0.0       6.0       0.8       0.0       8.0
0.2       4
   X'' + 0.2*X' + 1.0*SIGN(2.0*X - X') = 0.0

0.0
0.0       0.1       80.0
5.5
```

Example 17: X'' + 0.2*X' + 1.0*SIGN(2.0*X - X') = 0.0

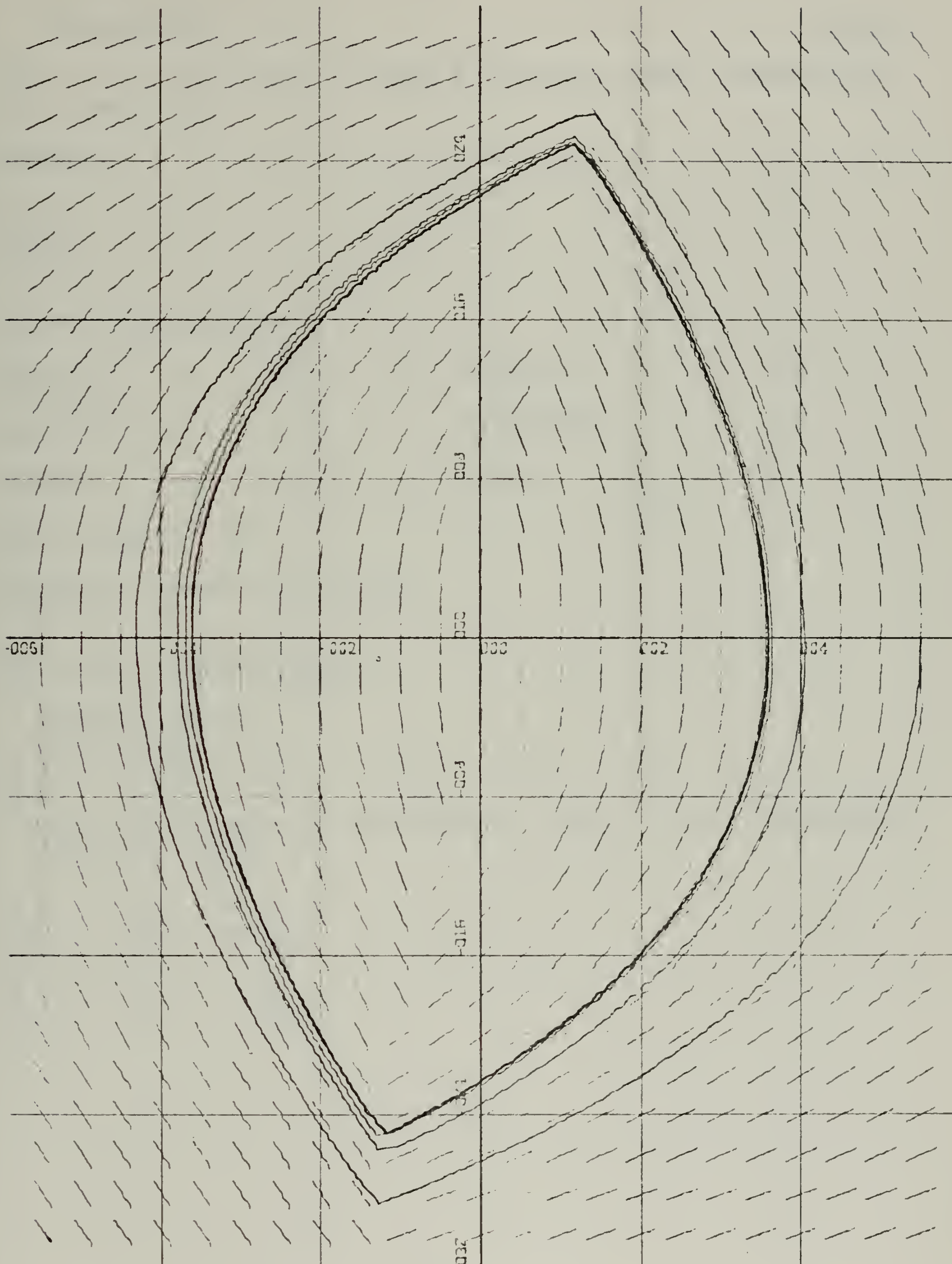

Fig. 5-17 X-SCALE = 2.0 units/inch

Y-SCALE = 0.8 units/inch

F. DEAD ZONE

Example 18:

IF(X. LT. -A. OR. X. GT. A) THEN X'' + B*X' + C*X = C*A*SIGN(X)
X'' + B*X' = 0. 0

WHERE:

A = 0. 3
B = 0. 2
C = 1. 0

PLOT PARAMETERS:

XSCALE	=	0. 5	YSCALE	=	0. 3
XCENTER	=	0. 0	YCENTER	=	0. 0
XSIZE	=	6. 0	YSIZE	=	8. 0

Size of slopes = 0. 2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
  NELSON, H. G.
  DEAD ZONE
0. 5      0. 0      6. 0      0. 3      0. 0      8. 0
0. 2      4
 IF(X. LT. -A. OR. X. GT. A) THEN X'' + B*X' + C*X = C*A*SIGN(X)
  X'' + B*X' = 0. 0

0. 3      0. 2      1. 0
0. 0      0. 1      50. 0
0. 0      1. 3
1
0. 0      0. 1      50. 0
1. 0
```

Example 18: Dead Zone

IF(X. LT. -A. OR. X. GT. A) THEN X'' + B*X' + C*X = C*A*SIGN(X)

X'' + B*X' = 0. 0

A = 0. 3, B = 0. 2, C = 1. 0

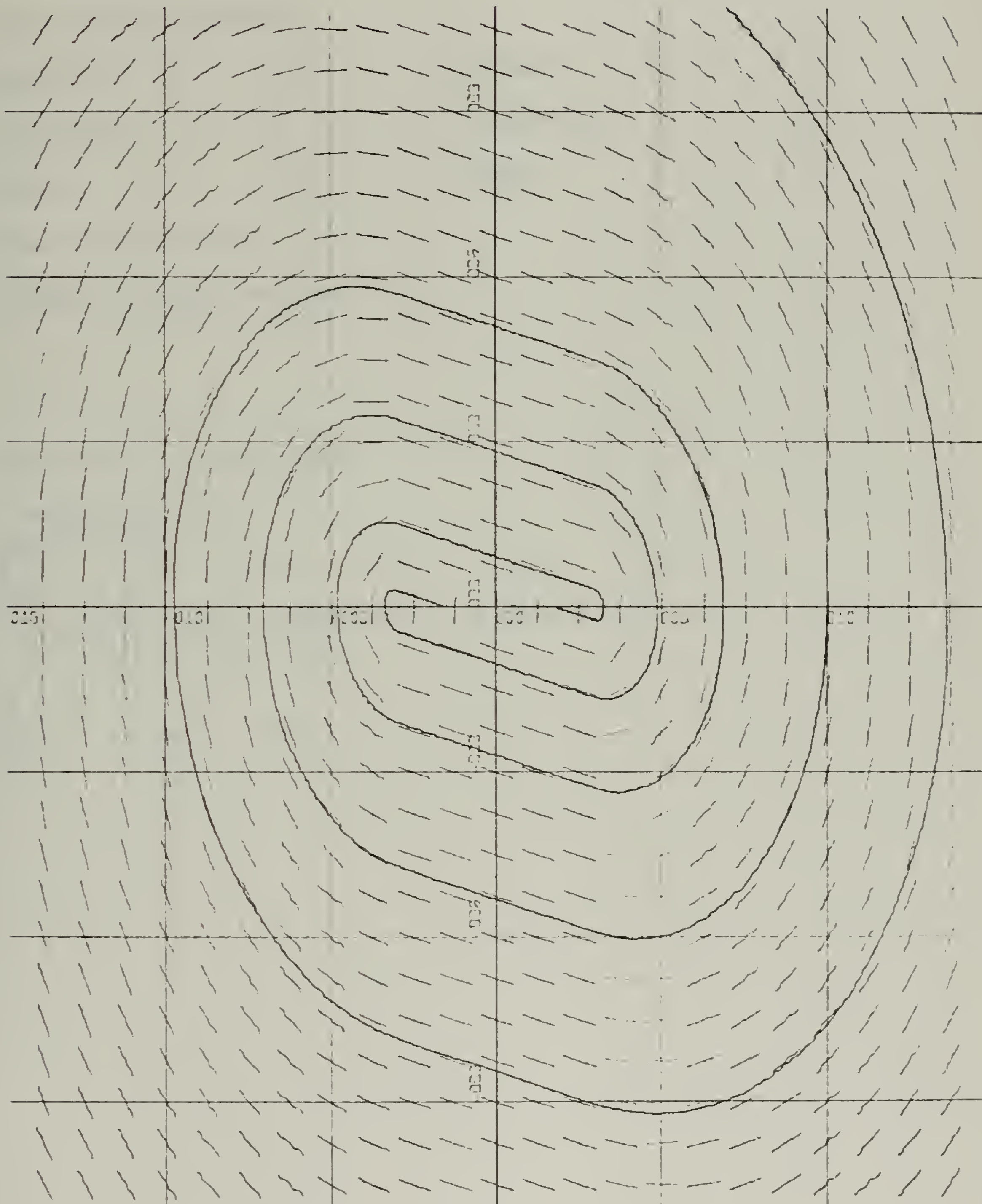

Fig. 5-18 X-SCALE = 0. 5 units/inch

Y-SCALE = 0. 3 units/inch

Example 19: Dead Zone

IF(ABS(X). LT. 2. 0) THEN X'' + 0. 2*X' = 0. 0
X'' + 0. 2*X' + X = 0. 0

PLOT PARAMETERS:

XSCALE	=	2. 0	YSCALE	=	2. 0
XCENTER	=	0. 0	YCENTER	=	0. 0
XSIZE	=	8. 0	YSIZE	=	6. 0

Size of slopes = 0. 2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
  NELSON, H. G.
DEAD ZONE
2.0      0.0      8.0      2.0      0.0      6.0
0.2      4
 IF(ABS(X).LT.2.0) THEN X'' + 0.2*X' = 0.0
 X'' + 0.2*X' + X = 0.0

0.0
0.0      0.05      35.0
6.0
```

Example 19: IF(ABS(X).LT.2.0) THEN X'' + 0.2*X' = 0.0
X'' + 0.2*X' + X = 0.0

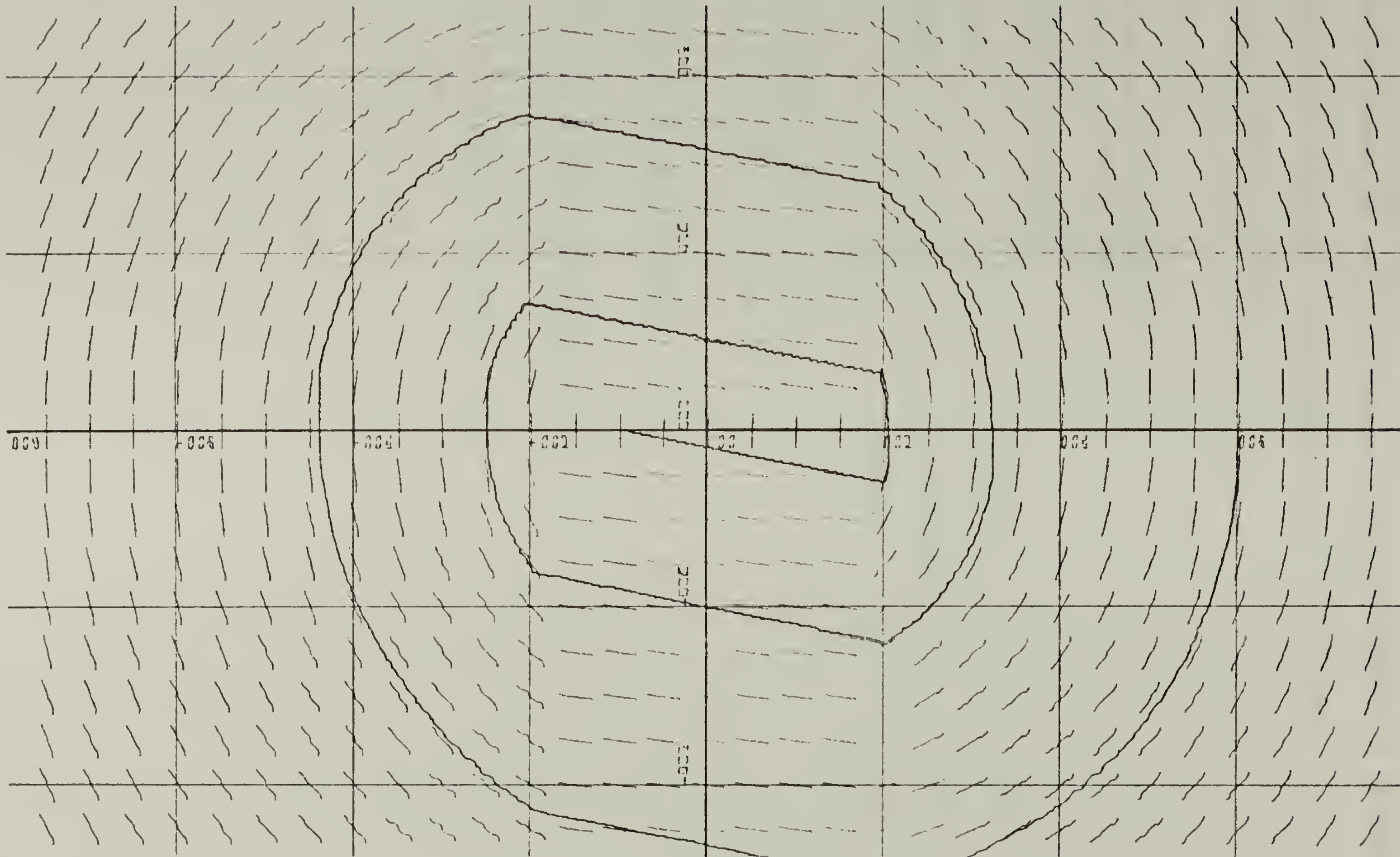

Fig. 5-19 X-SCALE = 2.0 units/inch

Y-SCALE = 2.0 units/inch

G. A SECOND ORDER LINEAR SYSTEM

Example 20:

The following three plots demonstrating the effect of various damping were the result of the input cards listed below.

X'' + 2.0*A*B*X' + B**2*X = 0.0

Where:

A = is the natural frequency
B = is the damping factor

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	2.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	8.0	YSIZE	=	6.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
  NELSON, H. G.
  X'' + 2.0*A*B*X' + B**2*X = 0.0
2.0       0.0       8.0       2.0       0.0       6.0
0.2       4
  X'' + 2.0*A*B*X' + 8**2*X = 0.0

1.0       1.0
0.0       0.1       10.0
7.0
1
0.0       0.1       10.0
-7.0
1
0.0       0.1       15.0
3.5
1
0.0       0.1       10.0
-3.5
2
0.5       1.0
0.0       0.1       15.0
7.0
1
```

```
0.0     0.1     15.0
-7.0
1
0.0     0.1     15.0
3.5
1
0.0     0.1     15.0
-3.5
2
0.1     1.0
0.0     0.1     80.0
5.0
```

Example 20(a) X'' + 2.0*A*B*X' + B**2*X = 0.0
A = 1.0, B = 1.0

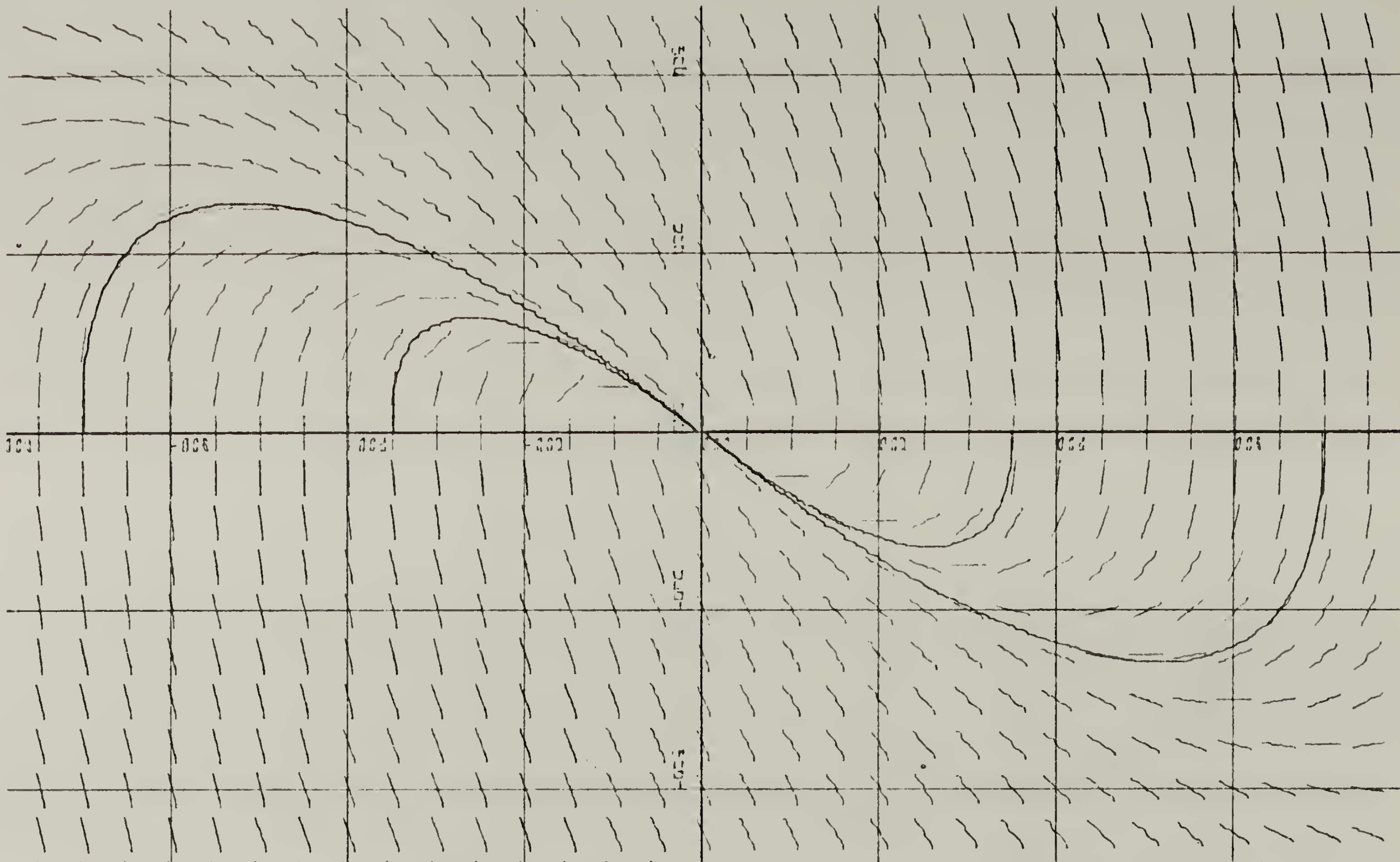

Fig. 5-20(a) X-SCALE = 2.0 units/inch

Y-SCALE = 2.0 units/inch

Example 20(b) X'' + 2.0*A*B*X' + B**2*X = 0.0

A = 0.5 B = 1.0

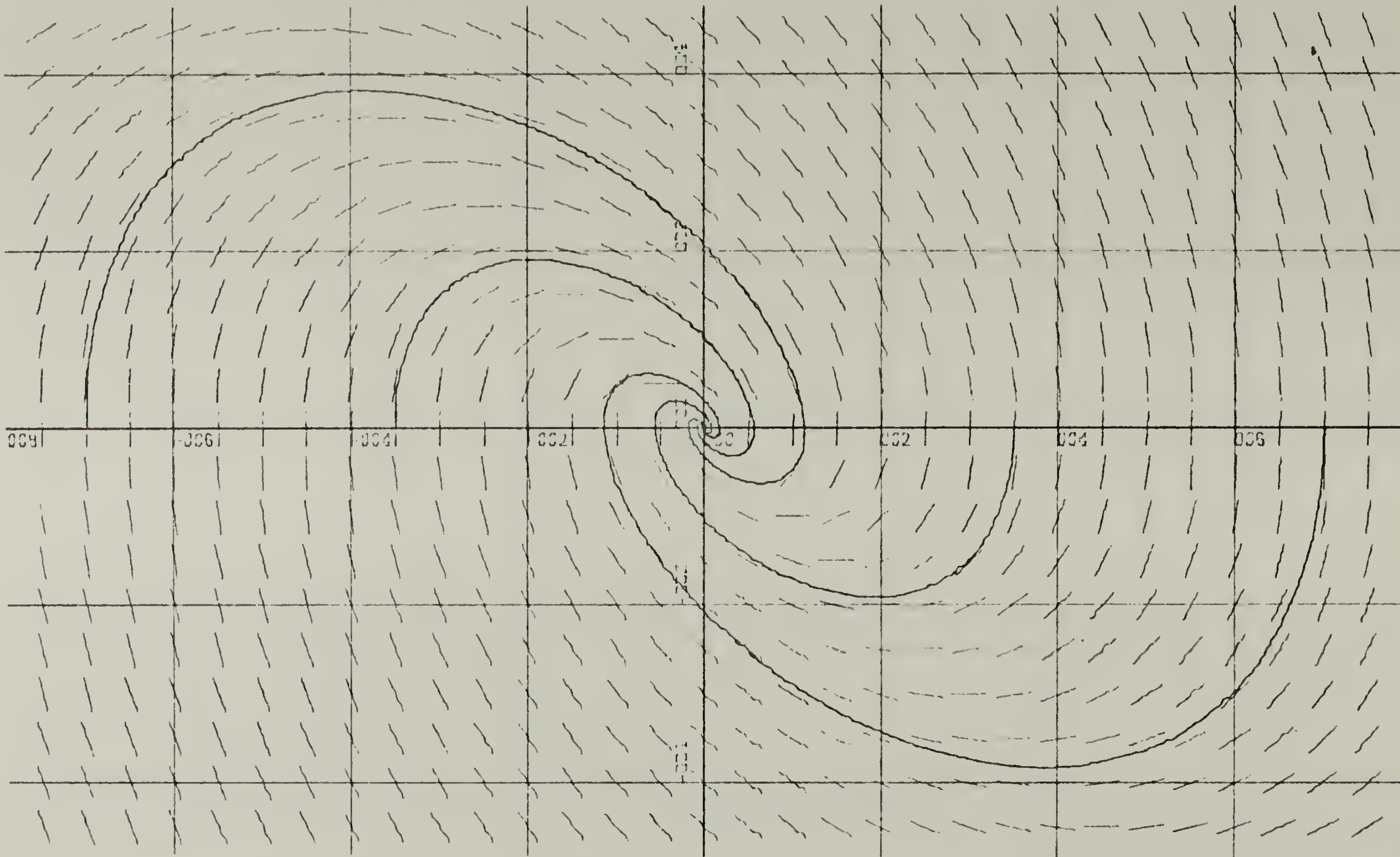

Fig. 5-20(b) X-SCALE = 2.0 units/inch

Y-SCALE = 2.0 units/inch

Example 20(c) X'' + 2.0*A*B*X' + B**2*X = 0.0
A = 0.1, B= 1.0

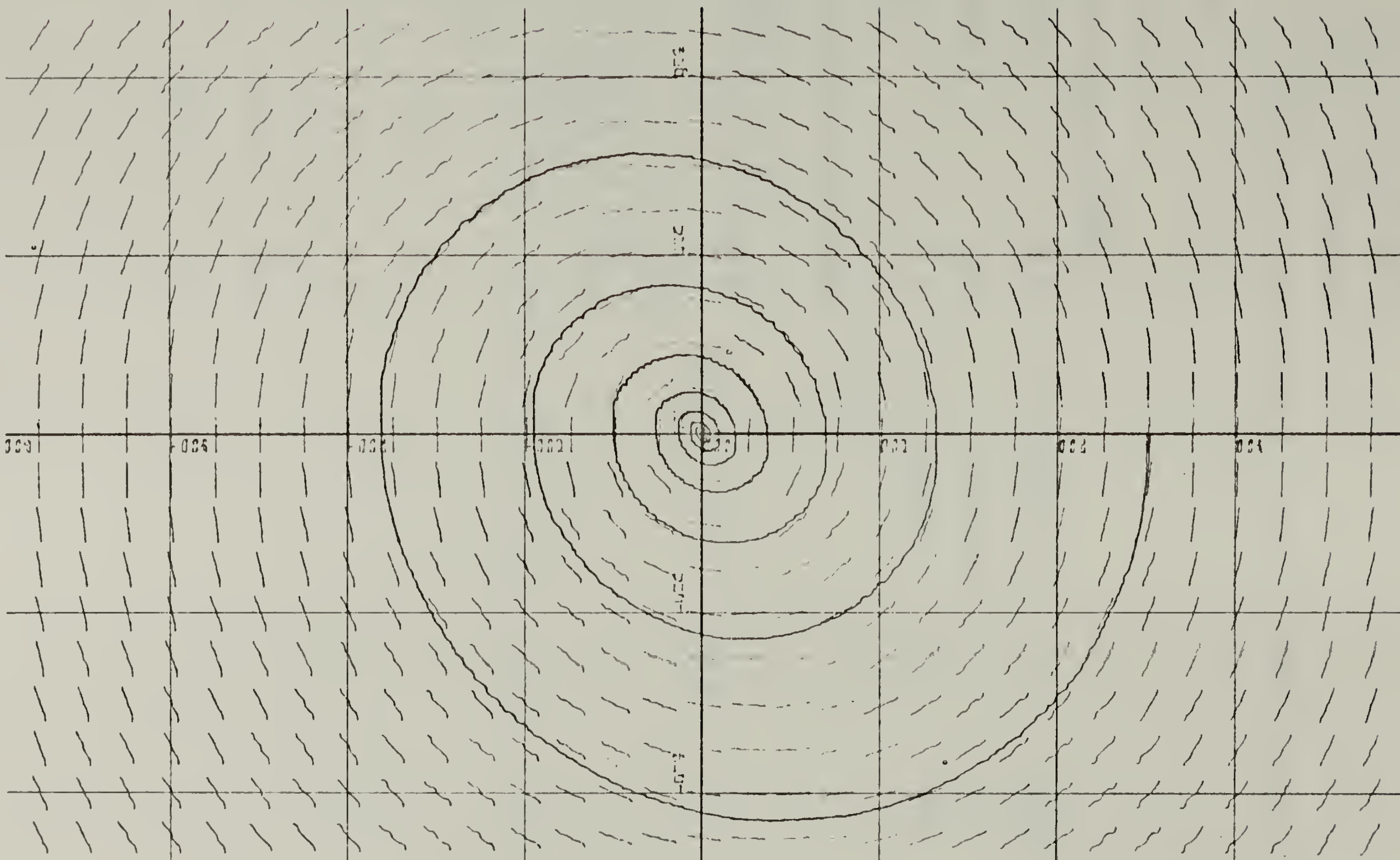

Fig. 5-20(c) X-SCALE = 2.0 units/inch
Y-SCALE = 2.0 units/inch

H. A FIFTH ORDER SYSTEM

Example 21:

```
IF(T  .GT.  0.0015) SKIP 3
V1 = 3000511.6 + A
V2 = 285120.0 + A*3511.0
V3 = A*38500.0
X''''' + 122.2*X'''' + 50835.8*X''' + V1*X'' + V2*X' + V3*X = 0.0
```

WHERE:

G = 30212.0

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	20.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

THE INPUT CARDS USED:

```
 NELSON, H. G.
 FIFTH ORDER SYSTEM
2.0       0.0       6.0       20.0      0.0       8.0

 IF(T.GT. 0.0015) SKIP 3
 V1 = 3000511.6 + A
 V2 = 285120.0 + A*3511.0
 V3 = A*38500.0
 X''''' + 122.2*X'''' + 50835.8*X''' + V*X'' + V2*X'  + V3*X = 0.0

30212.0
0.0       0.001       0.4
-3.0
1
0.0       0.001       0.4
2.5
1
0.0       0.001       0.4
-5.1
1
0.0       0.001       0.4
4.6
```

Example 21: A FIFTH ORDER SYSTEM
IF(T .GT. 0. 0015) SKIP 3
V1 = 3000511. 6 + A
V2 = 285120. 0 + A*3511. 0
V3 = A*38500. 0
X''''' + 122. 2*X'''' + 50835. 8*X''' + V1*X'' + V2*X' + V3*X = 0. 0

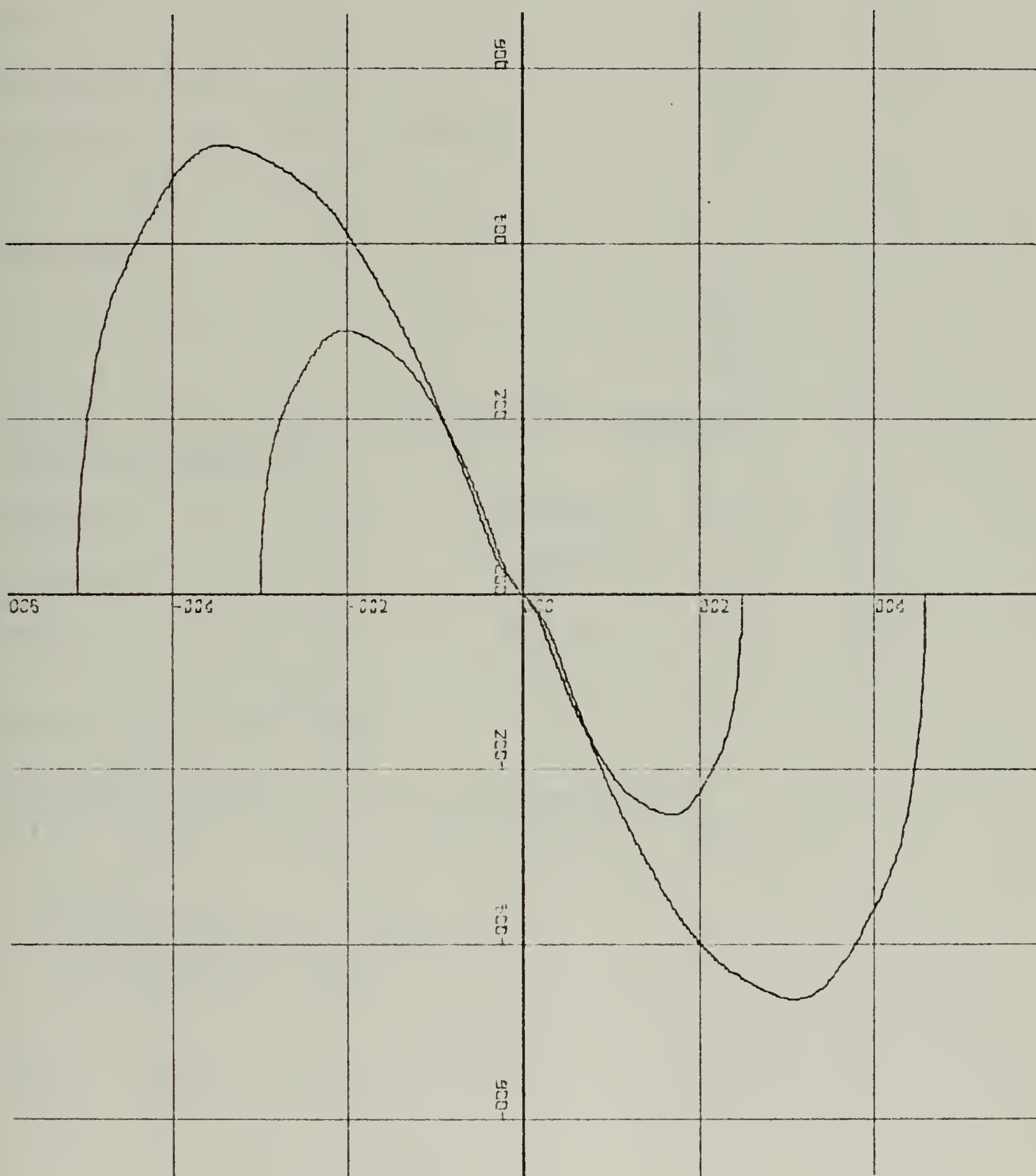

Fig. 5-21 X-SCALE = 2. 0 units/inch

Y-SCALE = 20. 0 units/inch

I. HYSTERESIS AND DEAD ZONE

Example 22:

RELAY WITH DEAD ZONE AND HYSTERESIS

THE SYSTEM DESCRIPTION EQUATIONS AND/OR PROGRAM:

V3 = T

IF(ABS(X) .LE. A)V1 = 0.0

IF(ABS(X) .GT. B)V1 = SIGN(X)

X'' + 0.2*X' + V1 = 0.0

WHERE:

A = 0.8

B = 1.2

The first statement is used to suppress the slopes.
PLOT PARAMETERS:

XSCALE	=	1.0	YSCALE	=	0.5
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

THE INPUT CARDS USED:

```
NELSON, H. G.
 RELAY WITH DEAD ZONE AND HYSTERISES
1.0       0.0       6.0       0.5       0.0       8.0

  V3 = T
  IF(ABS(X).LE.A) V1 = 0.0
  IF(ABS(X).GT.B) V1 = SIGN(X)
  X'' + 0.2*X' + V1 = 0.0

0.8       1.2
0.0       0.04      30.0
-2.8
```

Example 22: RELAY WITH DEAD ZONE AND HYSTERESIS

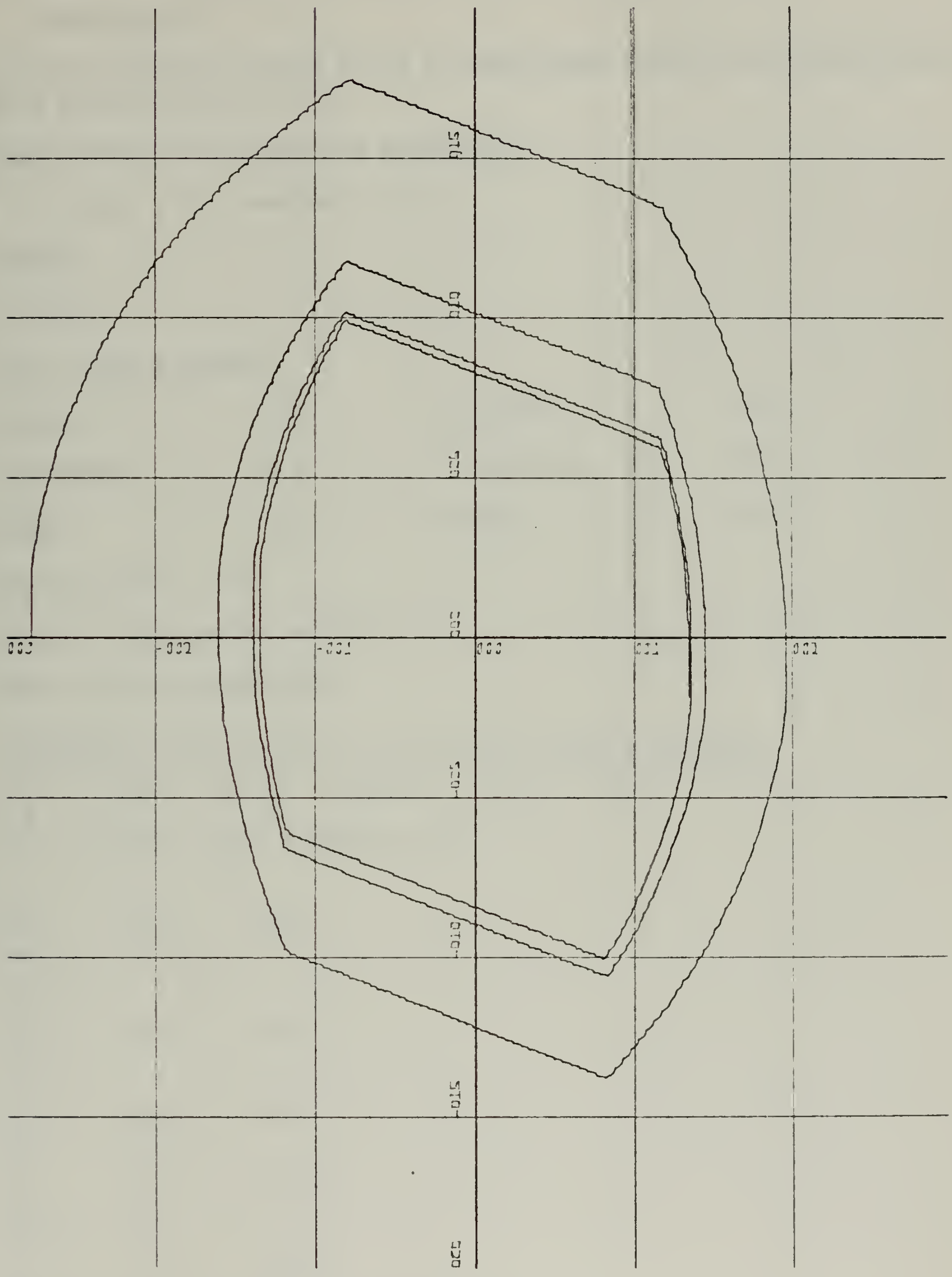

Fig. 5-22 X-SCALE = 1.0 units/inch

Y-SCALE - 0.5 units/inch

J. NON-LINEAR RESTORING FORCES

Example 23:

SECOND ORDER SYSTEM WITH A RESTORING FORCE CORRESPONDING TO A NON-LINEAR SPRING

THE SYSTEM DESCRIPTION EQUATION:

X'' + 0.9*X' + (X + A*X**3) = 0.0

WHERE:

A = 0.2

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	2.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

Size of slopes = 0.2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
 NELSON, H. G.
 SECOND ORDER SYSTEM WITH NON-LINEAR SPRING
2.0      0.0      6.0      2.0      0.0      8.0
0.2      4
 X'' + 0.9*X' + (X + A*X**3) = 0.0

0.2
0.0      0.02     15.0
-4.0
2
-0.1
0.0      0.02     15.0
-3.0
1
0.0      0.02     15.0
1.0      -7.0
1
0.0      0.02     15.0
-5.5     5.0
1
0.0      0.02     15.0
-5.2     5.0
1
0.0      0.02     15.0
5.0      -5.0
```

Example 23(a): X'' + 0. 9*X' + (X + A*X**3) = 0. 0

A = -0. 1 (gives a "soft" spring)

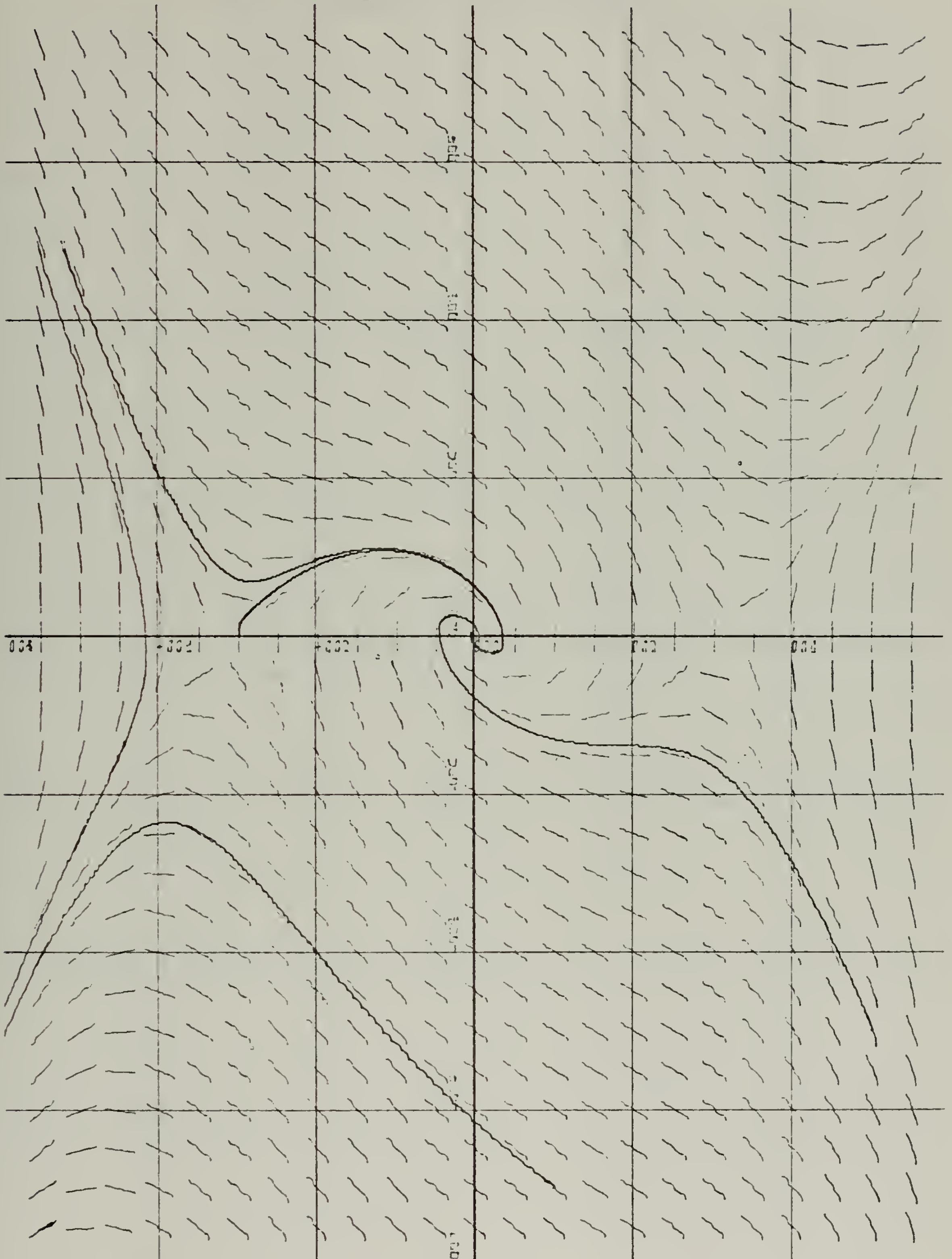

Fig. 5-23(a) X-SCALE = 2. 0 units/inch

Y-SCALE = 2. 0 units/inch

Example 23(b): X'' + 0.9*X' + (X + A*X**3) = 0.0

A = 0.2 (gives a "hard" spring)

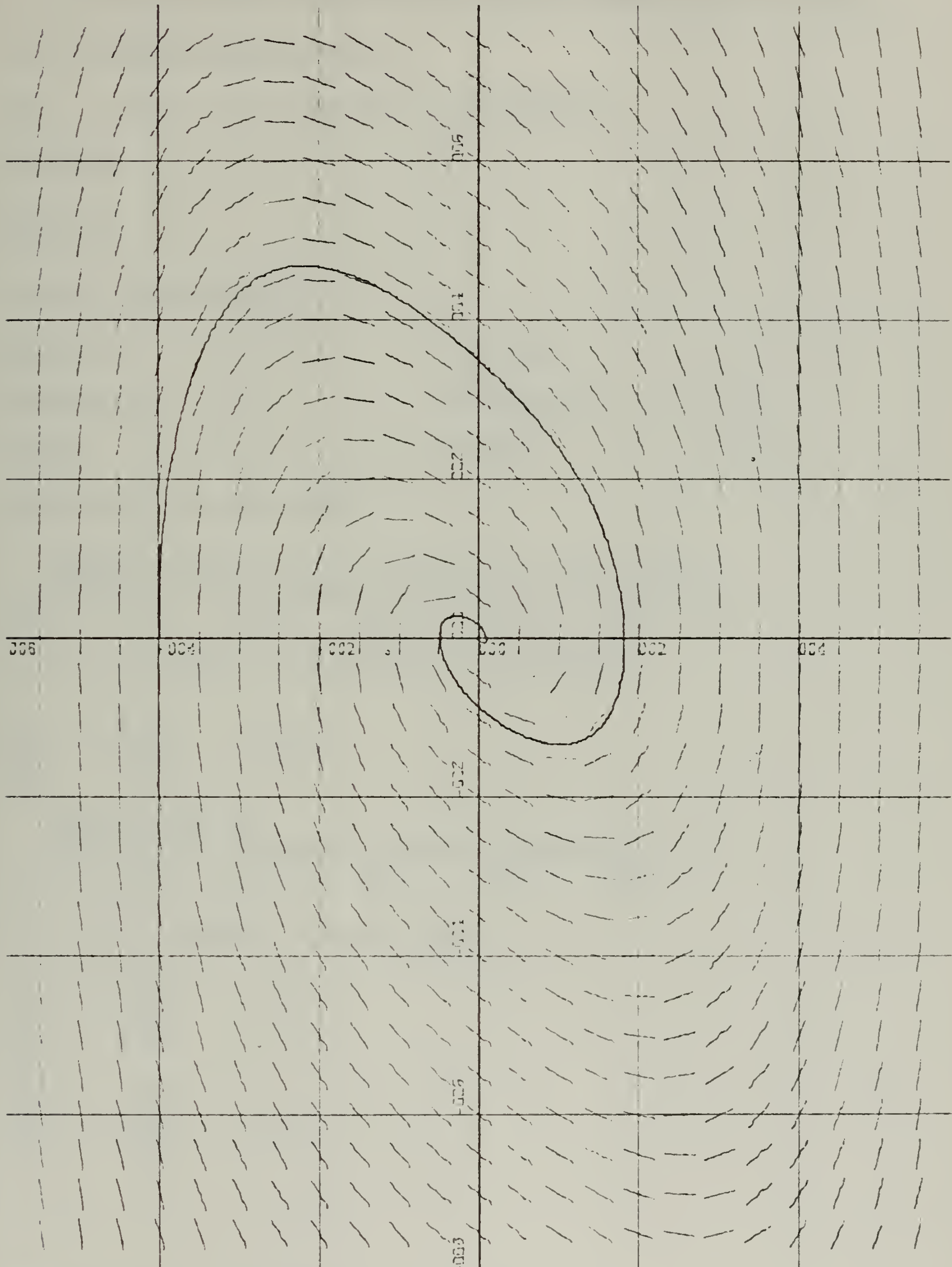

Fig. 5-23(b) X-SCALE = 2.0 units/inch

Y-SCALE = 2.0 units/inch

Example 24:

A FORCED SYSTEM WITH A NON-LINEAR RESTORING FORCE

THE SYSTEM DESCRIPTION:

X'' + 0.9*X' + (X + 0.2*X**3) = A*COS(B*T)

WHERE:

A = 3.0
B = 0.7

PLOT PARAMETERS:

XSCALE	=	1.0	YSCALE	=	1.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

THE INPUT CARDS USED:

```
  NELSON, H. G.
  FORCED NON-LINEAR SECOND ORDER SYSTEM
1.0    0.0     6.0     1.0     0.0     8.0

  X'' + 0.9*X' + (X + 0.2*X**3) = A*COS(B*T)

3.0    0.7
0.0    0.04    30.0
0.0    0.0
3
  NELSON, H. G.
  FORCED NON-LINEAR SECOND ORDER SYSTEM
2.0    0.0     6.0     2.0     0.0     8.0

X'' + (X + 0.2*X**3) = A*COS(B*T)

5.0    0.8
0.0    0.04    30.0
0.0    0.0
2
5.0    1.0
0.0    0.04    30.0
0.0    0.0
```

Example 24(a): X'' + 0.9*X' + (X + 0.2*X**3) = A*COS(B*T)
A = 3.0, B = 0.7

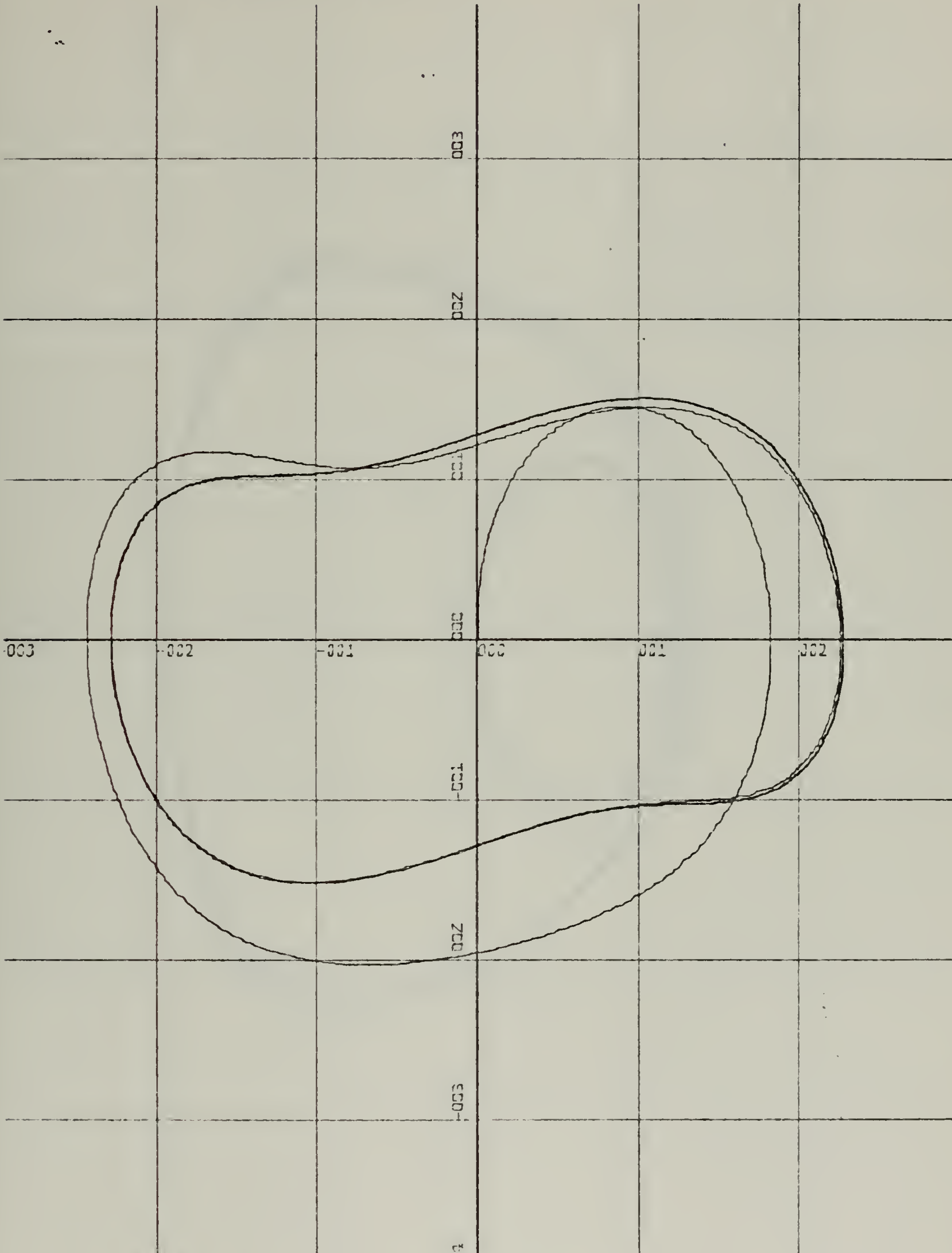

Fig. 5-24(a) X-SCALE = 1.0 units/inch

Y-SCALE = 1.0 units/inch

Example 24(b): X'' + 0. 0*X' + (X + 0. 2*X**3) = A*COS(B*T)

A = 5. 0, B = 0. 8

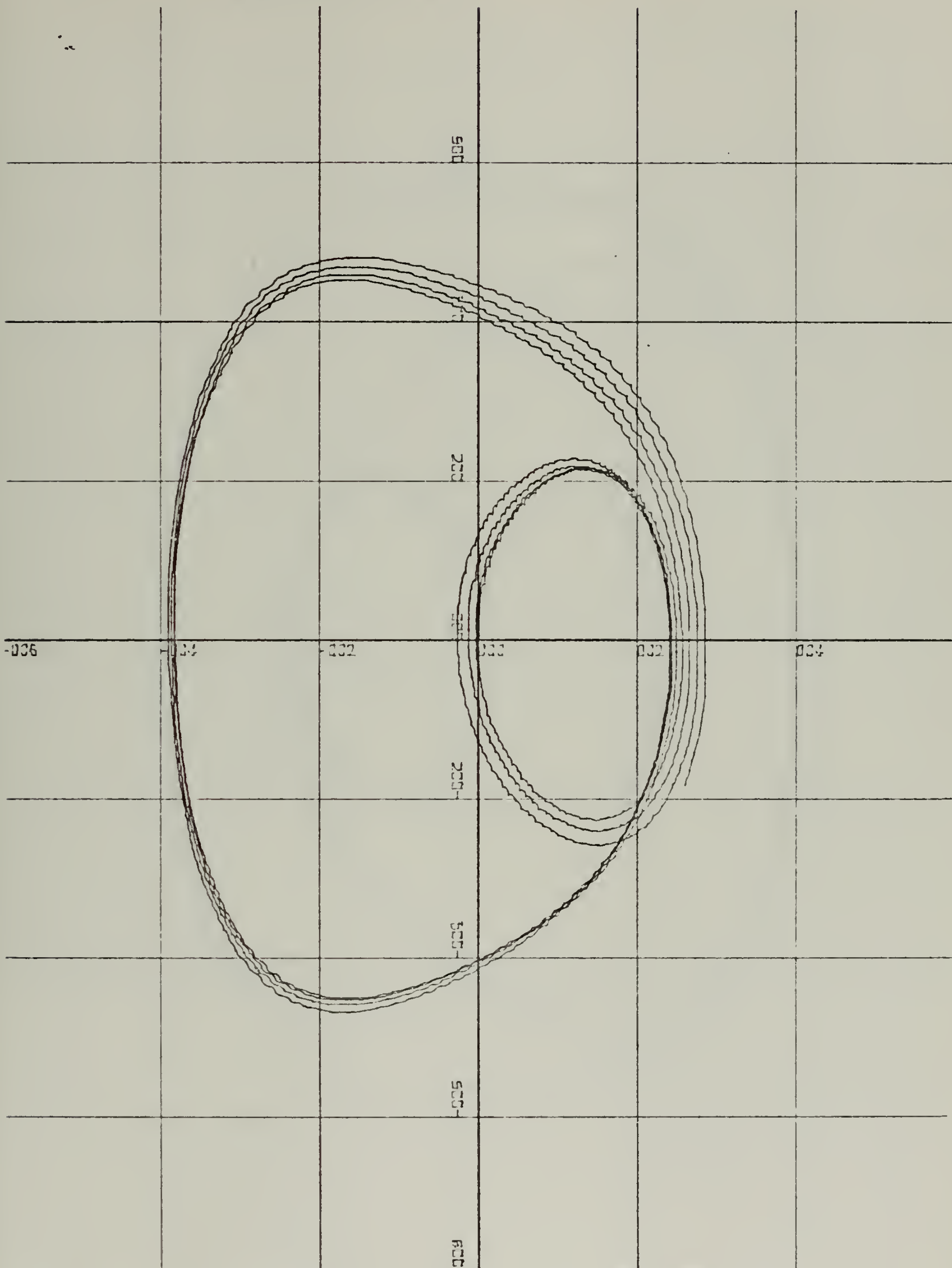

Fig. 5-24(b) X-SCALE = 1. 0 units/inch

Y-SCALE = 1. 0 units/inch

Example 24(c): X'' + 0. 0*X' + (X + 0. 2*X**3) = A*COS(B*T)
A = 5. 0, B = 1. 0

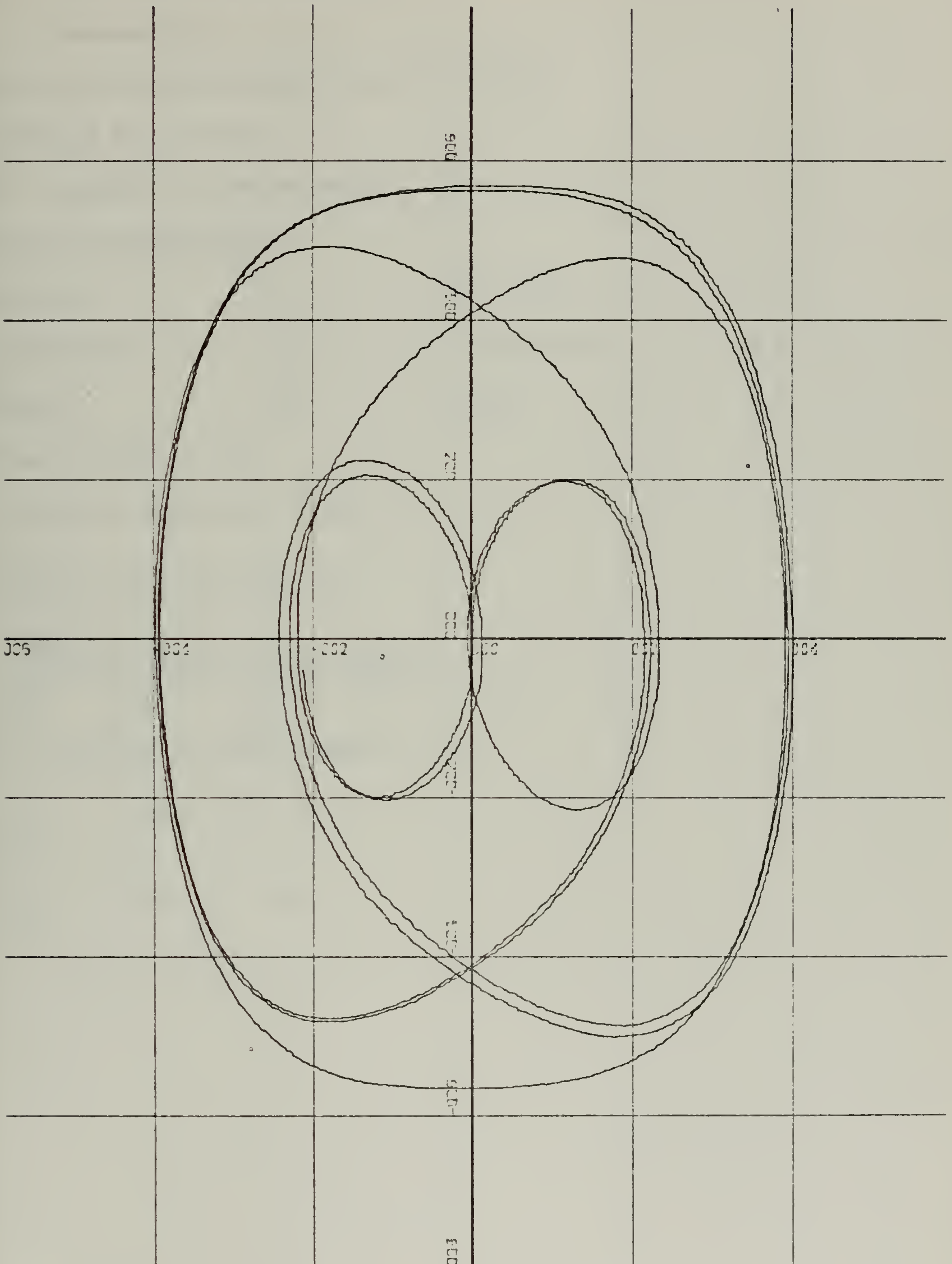

Fig. 5-24(c) X-SCALE = 1. 0 units/inch

Y-SCALE = 1. 0 units/inch

K. BANG-BANG SYSTEMS

Example 25.

A SECOND ORDER BANG-BANG SYSTEM

SYSTEM EQUATION:

X'' + SIGN(X + 0. 5*X'*ABS(X')) = 0. 0

PLOT PARAMETERS:

XSCALE	=	2. 0	YSCALE	=	1. 0
XCENTER	=	0. 0	Y CENTER	=	0. 0
XSIZE	=	8. 0	YSIZE	=	6. 0

Size of slopes = 0. 2

Number of slopes per inch = 4

THE INPUT CARDS USED:

```
 NELSON, H. G.
 SECOND ORDER BANG-BANG
2.0     0.0     8.0     1.0     0.0     6.0
0.2     4
X'' + SIGN(X + 0.5*X'*ABS(X')) = 0.0

0.0     0.02    15.0
-7.0
1
0.0     0.02    15.0
7.0
```

Example 25: A SECOND ORDER BANG-BANG SYSTEM

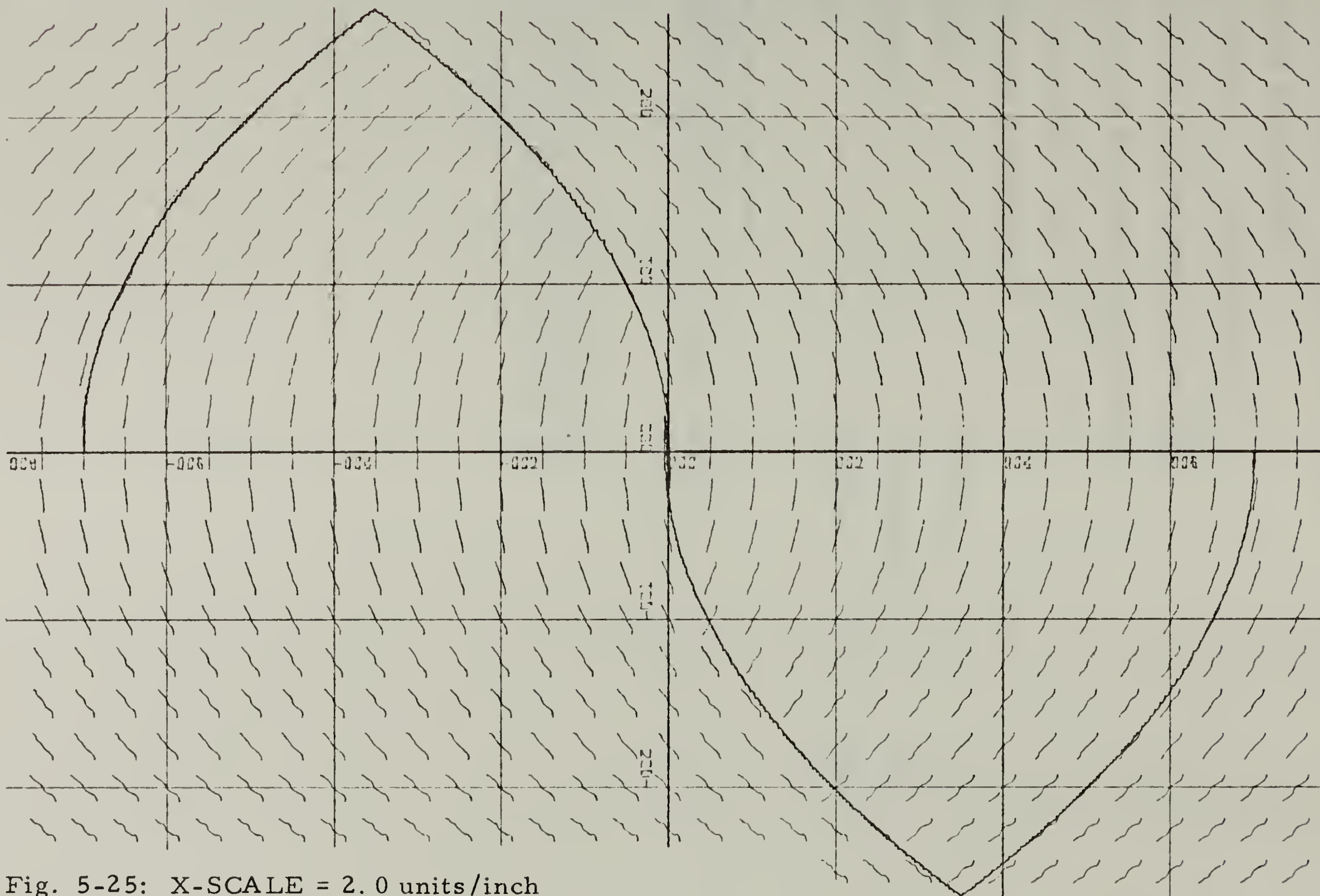

Fig. 5-25: X-SCALE = 2. 0 units/inch

Y-SCALE = 1. 0 units/inch

Example 26.

A THIRD ORDER BANG-BANG SYSTEM

SYSTEM EQUATIONS:

V1 = SIGN(X' + 0.5*X''*ABS(X''))

V2 = X + (1./3.)*X''**3 + V1*X''*X'

V3 = V1 *(0.5*X''**2 + V1*X')**1.5

X''' + SIGN(V2 + V3) = 0.0

PLOT PARAMETERS:

XSCALE	=	2.0	YSCALE	=	1.0
XCENTER	=	0.0	YCENTER	=	0.0
XSIZE	=	6.0	YSIZE	=	8.0

The input cards used:

```
 NELSON, H. G.
 THIRD ORDER BANG-BANG SYSTEM

2.0     0.0     6.0     1.0     0.0     8.0

V1 = SIGN(X' + 0.5*X''*ABS(X''))
V2 = X + (1./3.)*X''**3 + V1*X''*X
V3 = V1*(0.5*X''**2 + V1*X')**1.5
X''' + SIGN(V2 + V3) = 0.0
```

Example 26: A THIRD ORDER BANG-BANG SYSTEM

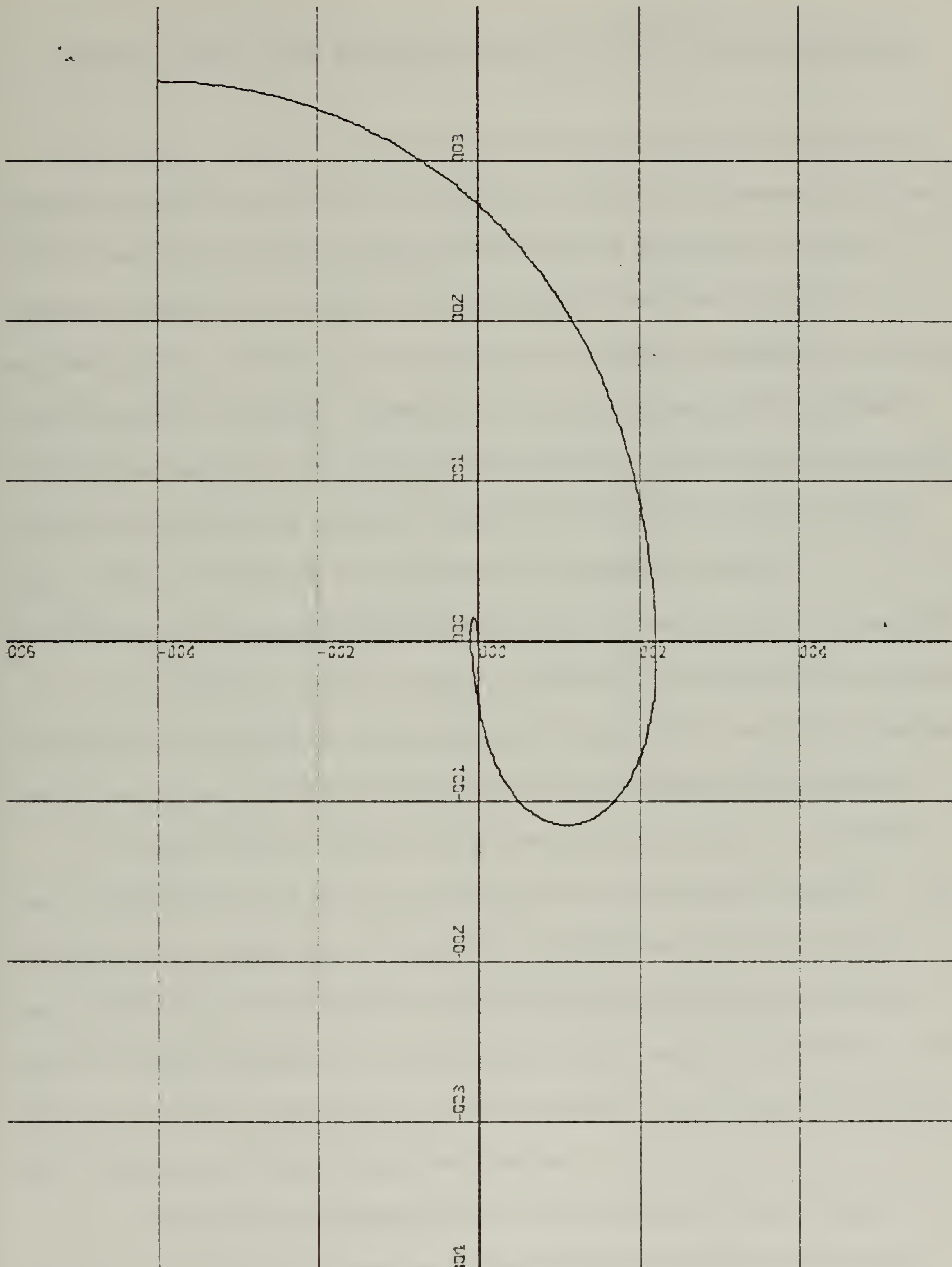

Fig. 5-26 X-SCALE = 2.0 units/inch

Y-SCALE = 1.0 units/inch

VI. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

This thesis has introduced an easy to use, flexible, and powerful analysis tool for the study of non-linear ordinary differential equations. The primary application emphasis has been in relation to control system problems for which many examples have been presented. It has been shown that the use of an automatic built-in compiler coupled with interactive graphics result in a very useful analysis program. For a large number of quite diverse examples it has been demonstrated in this thesis that one can very adequately specify a problem using data cards or by typing on an interactive graphics screen.

This study has suggested several basic ideas for further research.

1. An extension of the program following the development presented in Chapter II would give the program the capability of solving systems described by several simultaneous non-linear differential equations.

2. If one were concerned with execution efficiency, it would be very worthwhile to rewrite the interpreter in assembly language. The interpreter is called four times for each solution point and once for each slope line. Thus the slope generation and solution generation times could be speeded up considerably with a faster interpreter. The other parts of the program are used relatively infrequently and therefore efficiency is not an important factor.

3. It would be worthwhile to provide a choice of integration schemes to the user in addition to the present fourth order Runge-Kutta.

4. A built-in compiler could be used in several other applications. One of the very obvious areas is that of optimization. For example, a very useful directed search algorithm called "DIRECT" is available on the XDS 9300. The function to be minimized is incorporated into a function subroutine which must be supplied by the user. A modification of the compiler and interpreter presented in this thesis could be used as a very powerful interface for describing the function and its boundaries.

5. It would be very worthwhile developing methods to assist in the analysis of systems higher than second order. In general, the presentation of the grid of slopes is not worthwhile for systems of higher than second order since slope lines are in a space which has a dimension equal to the order of the differential equation.

APPENDIX A

REQUIRED JOB CONTROL LANGUAGE FOR BATCH VERSION

1. When using the compiled binary object deck use the following procedure:

```
//   JOB CARD       -Green-
//   EXEC FORTLG, REGION. GO=160K
//LINK. SYSIN DD *

     BINARY OBJECT DECK

//GO. FT06F001  DD  SYSOUT=A, SPACE=(CYL, (6))
//GO. SYSPLOT DD SYSOUT=C, SPACE=(TRK, (1, 10))
//GO. SYSPLOTS DD UNIT=SYSDA, SPACE=(TRK, (2, 6))
//GO. SYSIN DD *

     DATA

/*          -Orange-
```

2. If using the source deck use the following procedure:

```
//   JOB  CARD      -Green-
//   EXEC FORTCLGP, REGION. FORT=150K, REGION. GO=160K
//FORT. SYSPRINT DD SYSOUT=A, SPACE=(CYL, (1, 1))
//FORT. SYSIN DD *

      FORTRAN SOURCE PROGRAM

/*
//GO. FT06F001 DD SYSOUT=A, SPACE(CYL, (6))
//GO. SYSPLOT DD SYSOUT=C, SPACE=(TRK, (1, 10))
//GO. SYSPLOTS DD UNIT=SYSDA, SPACE=(TRK, (2, 6))
//GO. SYSIN DD *

      DATA

/*          - Orange-
```

APPENDIX B

1. Required control cards for Interactive Graphics Version.

BOOT CARD

PATCH DECK

```
Δ AGT
Δ JOB
Δ ASSIGN BI =MTOA
Δ ASSIGN GO =MTOA
Δ LOAD XM,  MAP
Δ SEG MAIN-(SINIT,  SSETUP2, SCOMP, (SGRID-SINTER), (SSOLVE-
Δ DATA                                                SINTER))
```

PROGRAM A

LISTING OF BATCH VERSION

```
C
C   THE DATA CARDS SHOULD BE ARRANGED AS FOLLOWS:
C
C     CARDS 1-2       PLOT TITLE  (6A8) FORMAT
C
C           3         X-SCALE (UNITS/INCH),
C                     X-CENTER (COORDINATE OF X-CENTER)
C                     X-SIZE (INCHES)
C                     Y-SCALE (AS ABOVE)
C                     Y-CENTER    "
C                     Y-SIZE      "
C                     THESE SIX ITEMS ARE PLACED ON ONE CARD IN THE
C                     ABOVE ORDER USING A 6F10.0 FORMAT.
C                     IF (X-SIZE .GT. 9.0) THEN THE PLOT IS ROTATED
C                     CCW ON THE PAPER BY A QUARTER TURN.  DUE
C                     TO PAPER SIZE ONLY ONE OF THE DIMENSIONS MAY
C                     EXCEED 9.0 INCHES.
C
C           4         SIZE OF SLOPE MARKERS(INCHES, 0.2 WORKS NICELY),
C                     NUMBER PER INCH(3 RECOMENDED)
C                                                IN (F10.4,I1) FORMAT
C
C           5,6,7,8,...  EQUATIONS TO BE USED: 80A1 FORMAT
C
C                     NOTE:
C                        1. THERE MAY BE MANY INSTRUCTIONS PER LINE,
C                           EACH MUST BE SEPARATED BY A SEMICOLON
C                        2.
C                        3.
C
C     THEN FOLLOW FOUR CARDS PER SOLUTION DESIRED AS DESCRIBED BELOW:
C
C           -FIRST-     VALUES OF A,B,C,D,E,F,G,H (8F10.4) FORMAT
C           -SECOND-    INTEGRATION INFO(INITIAL TIME, TIME STEP,
C               AND FINAL TIME) (900 SOLUTION STEPS MAX),(3F10.4) FORMAT
C           -THIRD-     INITIAL CONDITIONS FOR X, X', X'', . . .
C                                         IN (8F10.4) FORMAT
C
C           -FOURTH-    A "WHATS NEXT CARD" USE THE FOLLOWING CODE
C                         0  - THERE ARE NO REQUESTS FOLLOWING
C
C                         1  -  THE FOLLOWING ARE THE SECOND AND THIRD
C                               CARDS WITH NEW INTEGRATION AND INITIAL
C                               CONDITION INFORMATION ONLY.  I WANT
C                               THIS NEW SOLUTION TO BE PLOTTED ON THE
C                               PRESENT GRAPH.
```

```
C                              2  -  THE FOLLOWING IS ANOTHER FOUR CARDS
C                                       WITH DATA FOR ANOTHER SOLUTION OF
C                                       THE PRESENT SYSTEM.
C                              3  -  THE FOLLOWING CARD STARTS A COMPLETELY
C                                       NEW PROBLEM STARTING WITH CARD NUMBER
C                                       ONE.
C
C     A SAMPLE INPUT DECK FOLLOWS(PLEASE IGNORE THE "C" IN COLUMN
C        ONE AND THE SEQUENCE NUMBERING):
C
C
C
C   NELSON, H. G.    PENDULUM
C   X'' + A*X' + SIN(X) = 0
C2.0         0.0         8.0         2.0         0.0         6.0
C0.1         4
C   X'' + A*X' + SIN(X) = 0
C
C0.3
C0.0          0.05        25.0
C-7.0         3.0
C1
C0.0          0.05        25.0
C-7.0         5.0
C1
C0.0          0.05        25.0
C-7.0         2.0
C1
C0.0          0.05        25.0
C6.0         -5.0
C3
C   NELSON, H. G.    PENDULUM
C   X'' + A*X'*ABS(X') + SIN(X) = 0
C2.0         0.0         8.0         2.0         0.0         6.0
C0.2         4
C   X'' + A*X'*ABS(X') + SIN(X) = 0
C
C1.0
C0.0          0.03        20.0
C1.0          4.0
C1
C0.0          0.03        20.0
C-4.0         4.0
C1
C0.0          0.03        20.0
C-7.0         5.0
```

```
C1
C0.0          0.03       20.0
C5.0          -4.0
C1
C0.0          0.03       20.0
C2.0          4.0
C1
C0.0          0.03       20.0
C-2.5         -4.0
C3
C NELSON, H. G. 0902
C    SATURATED SERVO SYSTEM
C0.3          0.0        6.0        0.2        0.0        8.0
C0.2          4
C   IF(X.LT.(1.0-2.0)*0.2)  THEN X'' + 0.2*X' + X = 0.2      ;
C   IF(X.GT.0.2)  THEN X'' + 0.2*X' + X = -0.2
C   X'' + 0.2*X' + X = 0.0
C
C
C0.0           0.05       30.0
C0.7
C
C
C
      DOUBLE PRECISION ITITLE(12)
      DOUBLE PRECISION DEL
      REAL X(2),Y(2),LABEL/4H    /
      COMMON/STATUS/LSTAT
      DATA ITESTR/4H3   /
      WRITE(6,20)
   20 FORMAT('1')
   23 CALL INPUT1(XSCALE,YSCALE,XCENT,YCENT,SIZE,NO,ITITLE)
      CALL SETIME
      X(1) = XCENT
      X(2) = YCENT + XSCALE
      Y(1) = YCENT
      Y(2) = YCENT
      NEXT = 3
      CALL COMP
      IF(LSTAT.EQ.3) GO TO 31
   22 CALL    GRID(XSCALE,YSCALE,XCENT,YCENT,SIZE,NO,ITITLE)
   21 CALL INPUT2  (TI,DEL,TF,NEXT)
      IF(NEXT.EQ.1) GO TO 24
      CALL SLOPES(XSCALE,YSCALE,XCENT,YCENT,SIZE,NO,ITITLE)
      IF(LSTAT.EQ.3) GO TO 31
      DO 24 I=1,2
```

```
24 CALL SOLVE(TI,DEL,TF,XI,YI,ITITLE)
   IF(LSTAT.EQ.3) GO TO 31
   READ(5,10)NEXT
10 FORMAT(I1)
   CALL SETIME
   IF(NEXT.EQ.1) GO TO 21
C
C  TERMINATING PRESENT GRAPH WITH A DUMMY CALL TO DRAW
C
   CALL DRAW(2,X,Y,3,0,LABEL,ITITLE,0.,0.,0.,0.,0,0,0,0,0,LSAT)
   CALL GETIME(IET)
   ET = IET*0.00026/60.0
   WRITE(6,40) ET
40 FORMAT('0 EXECUTION TIME FOR THIS PROBLEM WAS:',F10.5)
   IF(NEXT) 13,13,14
14 GO TO (13,22,23),NEXT
31 CONTINUE
   READ(5,30) ITEST
30 FORMAT(A4)
   IF(ITEST.NE.ITESTR) GO TO 31
   WRITE(6,20)
   GO TO 23
13 STOP
   END
```

```
C
C
      SUBROUTINE INPUT1(XSCALE,YSCALE,XCENT,YCENT,SIZE,NO,ITITLE)
      DOUBLE PRECISION ITITLE(12)
      COMMON/PSIZE/NXSIZE,NYSIZE
      WRITE(6,50)
   50 FORMAT('1',//////////,' PLOT PARAMETERS:')
      READ(5,10)ITITLE
      WRITE(6,110)ITITLE
   10 FORMAT(6A8)
  110 FORMAT('0PLOT TITLE:',/,6A8,/,6A8)
      READ(5,20)XSCALE,XCENT,XSIZE,YSCALE,YCENT,YSIZE
      CALL SNORM(XSCALE)
      CALL SNORM(YSCALE)
      IF(XSIZE.GT.15.0) XSIZE = 15.0
      IF(YSIZE.GT.15.0) YSIZE = 15.0
      NXSIZE = XSIZE + 0.5
      NYSIZE = YSIZE + 0.5
      WRITE(6,120)XSCALE,YSCALE,XCENT,YCENT,XSIZE,YSIZE
  120 FORMAT('0XSCALE     = ',F11.4,12X,'YSCALE     = ',F11.4,/,
     *       '0XCENTER    = ',F11.4,12X,'YCENTER    = ',F11.4,/,
     *       '0XSIZE      = ',F11.4,12X,'YSIZE      = ',F11.4)
   20 FORMAT(8F10.3)
      READ(5,40)SIZE,NO
      WRITE(6,140)SIZE,NO
   40 FORMAT(F10.4,I1)
  140 FORMAT('0SIZE OF SLOPES =',F5.1,/
     */,' NUMBER OF SLOPES PER INCH = ',I2)
      RETURN
      END
```

```
C
      SUBROUTINE SNORM(SCALE)
      AN = 0.1
      N = 0
      IF(SCALE.GT.1.0) AN = 10.0
   21 IF(SCALE.GE.1.0.AND. SCALE.LT.10.0) GO TO 22
      N = N + 1
      SCALE = SCALE/AN
      GO TO 21
   22 NSCALE = SCALE + 0.5
      SCALE = NSCALE
      IF(N.EQ.0) RETURN
      DO 91 I=1,N
      SCALE = SCALE*AN
   91 CONTINUE
      RETURN
      END
```

```
C
      SUBROUTINE INPUT2(TI,DEL,TF,NEXT)
      REAL ZI(9)
      DOUBLE PRECISION DEL
      COMMON/ICON/ ZI
      COMMON/COEF/A,B,C,D,E,F,G,H
      ZI(9) = 0.0
      WRITE(6,20)
   20 FORMAT('1**** THE SOLUTION PARAMETER DATA CARDS FOLLOW: ****')
      IF(NEXT.EQ.1) GO TO 21
      READ(5,10)A,B,C,D,E,F,G,H
      WRITE(6,110)A,B,C,D,E,F,G,H
  110 FORMAT(' WHERE:',//,' A= ',F12.5,/,' B = ',F12.5,/,' C = ',F12.5,/
     *,      ' D = ',F12.5,/,' E = ',F12.5,/,' F = ',F12.5,/,
     *       ' G = ',F12.5,/,' H = ',F12.5,/,'1')
   21 READ(5,10) TI,DEL,TF
      WRITE(6,120) TI,DEL,TF
  120 FORMAT('0THE INITIAL TIME, TIME STEP, AND FINAL TIME ARE:'/3F15.5)
      READ(5,10) (ZI(I),I=1,8)
      WRITE(6,130) (ZI(I),I=1,8)
  130 FORMAT('0THE INITIAL CONDITIONS ARE:',/,8F12.5)
   10 FORMAT(8F10.4)
      RETURN
   99 STOP
      END
```

```
C
C
      SUBROUTINE GRID(XSCALE,YSCALE,XCENT,YCENT,SIZE,NO,ITITLE)
      DIMENSION X(10),Y(10)
      REAL LABEL/4H    /
      DOUBLE PRECISION ITITLE(12)
      COMMON/PSIZE/NXSIZE,NYSIZE
      CALL PLOTS
      CALL PLOT(-1.0E+50,0.,-3)
      CALL PLOT(1.0,0.0,-3)
      CALL PLOTE
      XMIN=XCENT-XSCALE*NXSIZE/2.
      XMAX=XCENT+XSCALE*NXSIZE/2.
      YMIN=YCENT-YSCALE*NYSIZE/2.
      YMAX=YCENT+YSCALE*NYSIZE/2.
      IYRIGH=(XCENT-XMIN)/XSCALE + 0.5
      IXUP=(YCENT-YMIN)/YSCALE + 0.5
C     WRITE(6,10)XMIN,XMAX,YMIN,YMAX,IYRIGH,IXUP
   10 FORMAT(' IN GRID, XMIN,XMAX,YMIN,YMAX ARE: ',4F10.3,2I10)
      X(1)=XMIN
      Y(1)=YMIN
      X(2)=X(1)
      Y(2)=YMAX
      X(3)=XMAX
      Y(3)=Y(2)
      X(4)=XMAX
      Y(4)=YMIN
      X(5)=X(1)
      Y(5)=Y(1)
      X(6)=XCENT
      Y(6)=YMIN
      X(7)=XCENT
      Y(7)=YMAX
      X(8)=XCENT
      Y(8)=YCENT
      X(9)=XMAX
      Y(9)=YCENT
      X(10)=XMIN
      Y(10)=YCENT
      IF(NXSIZE.GT.9) GO TO 21
      CALL DRAW(10,X,Y,1,0,LABEL,ITITLE,XSCALE,YSCALE,IXUP,IYRIGH,2,2,
     *NXSIZE,NYSIZE,1,LAST)
      RETURN
   21 DO 91 J = 1,10
   91 Y(J) = -Y(J)
      CALL DRAW(10,Y,X,1,0,LABEL,ITITLE,YSCALE,XSCALE,IYRIGH,IXUP,2,2,
```

```
*NXSIZE,NYSIZE,1,LAST)
 RETURN
 END
```

```
C
C
      SUBROUTINE SLOPES(XSCALE,YSCALE,XCENT,YCENT,SIZE,NO,ITITLE)
      DIMENSION X(2),Y(2)
      REAL LABEL(9)/9*4H    /
C     REAL LABEL(4)/' 1  ',' 2  ',' 3  ',' 4  '/
      REAL *8 ITITLE(12),T
      DOUBLE PRECISION Z(10), ZDOT(10)
      COMMON Z, ZDOT, T, L, N, LFUNCT
      COMMON/STATUS/LSTAT
      COMMON/PSIZE/NXSIZE,NYSIZE
      T = 0.0
      LFUNCT = 0
      XMIN=XCENT-XSCALE*NXSIZE/2.
      YMIN=YCENT-YSCALE*NYSIZE/2.
      NX= NXSIZE*NO-1
      NY= NYSIZE*NO-1
C     WRITE(6,20)XMIN,YMIN,NX,NY
   20 FORMAT(' XMIN,YMIN,NX,NY:', 2F15.5,2I10)
      DO 91 IX=1,NX
      Z(1)=XMIN+IX*XSCALE/NO
      DO 91 IY=1,NY
      Z(2)=YMIN+IY*YSCALE/NO
      CALL ZDVAL
      IF(LSTAT.EQ.3) RETURN
      IF(N.GT.2) RETURN
      IF(LFUNCT.EQ.1) GO TO 91
      IF(DABS(Z(2)).GT.1.E-20) GO TO 25
      THETA = 3.14159/2.
      GO TO 32
   25 SLOPE = (ZDOT(2)/YSCALE)/(Z(2)/XSCALE)
      THETA=ATAN(SLOPE)
   32 AMAGX = SIZE*XSCALE*COS(THETA)/2.
      AMAGY = SIZE*YSCALE*SIN(THETA)/2.
      X(1)=Z(1)+AMAGX
      X(2)=Z(1)-AMAGX
      Y(1)=Z(2)+AMAGY
      Y(2)=Z(2)-AMAGY
      IF(NXSIZE.GT.9) GO TO 21
      CALL DRAW(2,X,Y,2,0,LABEL(L),ITITLE,0.,0.,0.,0.,0,0,0,0,0,LAST)
      GO TO 91
   21 DO 92 J = 1,10
   92 Y(J) = -Y(J)
      CALL DRAW(2,Y,X,2,0,LABEL(L),ITITLE,0.,0.,0.,0.,0,0,0,0,0,LAST)
   91 CONTINUE
   99 RETURN
      END
```

```
C
C
      SUBROUTINE SOLVE(TI,DEL,TF,XI,YI,ITITLE)
      DIMENSION ZI(9), D(901,6), JXY(6)
      DOUBLE PRECISION Z(10),ZDOT(10),T,DEL,ITITLE(12)
      REAL LABEL/4H    /
      COMMON Z, ZDOT, T, L, N, LFUNCT
      COMMON/ICON/ ZI
      COMMON/STATUS/LSTAT
      COMMON/PSIZE/NXSIZE,NYSIZE
      WRITE(6,30)
   30 FORMAT('1AREA             TIME             X(1)             X(2)             X(
     *3)          X(4)             X(5)             X(6)             X(7)
     *  X(8) ')
      NT=0
      LR = 0
      MM= ((TF-TI)/DEL)/40
      T=TI
      D(1,1) = TI
      DO 91 J=1,5
      Z(J) = ZI(J)
      D(1,J+1) = ZI(J)
   91 CONTINUE
      I=1
   18 I=I+1
   12 CALL ZDVAL
      IF(LSTAT.EQ.3) RETURN
      S=RKLDEQ(N,Z,ZDOT,T,DEL,NT)
   50 FORMAT(6F15.5)
      IF(S-1.0)11,12,14
   11 STOP
   14 NP1 = N + 1
      D(I,1) = T
      DO 92 J=2,NP1
      D(I,J) = Z(J-1)
   92 CONTINUE
      CALL ZDVAL
      IF(LSTAT.EQ.3) RETURN
      IF(.NOT.((MOD(I,MM).EQ.0).OR.(L.NE.LR).OR.(T.GE.TF))) GO TO 21
      WRITE(6,20)L,T,(Z(J),J=1,N)
   20 FORMAT(I5,9F14.4)
   21 LR = L
      IF(900-I)15,15,16
   15 WRITE(6,10)T
   10 FORMAT(' SOLUTION TERMINATED AT TIME =',F10.5,
     *'   WITH 900 SOLUTION POINTS')
```

```
      GO TO 17
   16 IF(TF-T)17,17,18
   17 CONTINUE
      IF(NXSIZE.GT.9) GO TO 121
      CALL DRAW(I,D(1,2),D(1,3),2,0,LABEL,ITITLE,0.,0.,0.,0.,0,0,0,0,0,
     *LAST)
      GO TO 22
  121 DO 94 J = 1,I
      D(J,3) = -D(J,3)
   94 CONTINUE
      CALL DRAW(I,D(1,3),D(1,2),2,0,LABEL,ITITLE,0.,0.,0.,0.,0,0,0,0,0,
     *LAST)
   22 JXY(1) = 2
      JXY(2) = 3
      CALL VPLOT(D,JXY,I,901,1,0,0.0,0.0,0.0,0.0)
      WRITE(6,70)
   70 FORMAT(' THE ABOVE GRAPH IS A PHASE PLANE PLOT OF THE SECOND',
     *' DERIVATIVE VERSUS THE FIRST DERIVATIVE')
      JXY(1) = 1
      DO 93 J =2,NP1
      JXY(2) = J
      CALL VPLOT(D,JXY,I,901,1,0,0.0,0.0,0.0,0.0)
      JM1 = J - 1
      WRITE(6,80) JM1
   80 FORMAT('0THE ABOVE GRAPH IS A PLOT OF THE NUMBER ',I2,
     *' DERIVATIVE VERSUS TIME')
   93 CONTINUE
      RETURN
      END
```

```
C
C     ..................................................................
C        FORTRAN 4 VERSION OF RUNGE-KUTTA-GILL ROUTINE
C         X,Y,F,H  ARE DOUBLE-PRECISION. MAX N = 25
      FUNCTION RKLDEQ (N,Y,F,X,H,NT)
      DOUBLE PRECISION Y,F,X,H,Q,H1,H2,H3,H6
      DIMENSION Y(2), F(2), Q(25)
C
      NT = NT +1
      GO TO (1,2,3,4),NT
    1 H1 = H
      H2 = H1 * .5D0
      H3 = H1 * 2.D0
      H6 = H1/6.D0
      DO 11 J =1,N
   11 Q(J) = 0.D0
      A = .5D0
      X = X + H2
      GO TO 5
C
    2 A = .2928932188134525
      GO TO 5
C
    3 A = 1.7071067811865475
      X = X + H2
      GO TO 5
C
    4 DO 41 I = 1,N
   41 Y(I) = Y(I) + H6 * F(I) -Q(I)/3.D0
      NT = 0
      RKLDEQ =2.
      GO TO 6
C
    5 DO 51 L = 1,N
      Y(L) = Y(L) + A *(H * F(L) -Q(L))
   51 Q(L) = H3 * A *F(L) +(1.D0-3.D0*A) *Q(L)
      RKLDEQ =1.
C
    6 RETURN
      END
```

```
C       SUBROUTINE COMP
C
C    THE NUMERICAL CODES ASSIGNED TO THE VARIOUS OPERANDS
C    AND OPERATORS ARE LISTED BELOW
C
C      'Z(1)'    1     -1
C      'Z(2)'    2     -1
C      'Z(3)'    3     -1
C      'Z(4)'    4     -1
C      'Z(5)'    5     -1
C      'Z(6)'    6     -1
C      'Z(7)'    7     -1
C      'Z(8)'    8     -1
C      'Z(9)'    9     -1
C      '-1.0'    10    -1
C      'A'       11    -1
C      'B'       12    -1
C      'C'       13    -1
C      'D'       14    -1
C      'E'       15    -1
C      'F'       16    -1
C      'G'       17    -1
C      'H'       18    -1
C      'T'       19    -1
C
C      COMPILER ASSIGNED VARIABLES ARE Z(21) THRU Z(60)
C
C      USER ASSIGNED VARIABLES ARE V1 THRU V40  CORRESPONDING
C               TO INTERNAL DESIGNATIONS Z(61) THRU Z(100)
C
C       PLUS    121    7
C       MINUS   122    7
C       DIV     123    8
C       MULT    124    8
C       PWR     125    9
C       PIPWR   126    9
C       PNPWR   127    9
C
C      'SIN'    131    10
C      'COS'    132    10
C      'TAN'    133    10
C      'ABS'    134    10
C      'EXP'    135    10
C      'LN'     136    10
```

```
C     'LOG'     137    10
C     'INT'     138    10
C     'SIGN'    139    10
C
C     'THEN'    141
C     '('       142    1
C     ')'       143    2
C     APOST.    144
C     IF        145    11
C     TRUE      146    -1
C     SCOLON    147    0
C     EQUAL     148    0
C
C     '.OR.'    151    3
C     '.AND.'   152    4
C     '.NOT.'   153    5
C     '.GT.'    154    6
C     '.GE.'    155    6
C     '.LT.'    156    6
C     '.LE.'    157    6
C     '.EQ.'    158    6
C     '.NE.'    159    6
C
C     SKIPS ARE NUMBERED 161 - 169, LAST DIGIT INDICATES NO. OF SKIPS
C     D.E. EQUATION DEGREES ARE 171-179, LAST DIGIT INDICATES EQUATION
C     D. E. EQUATION NUMBER (ASSIGNED CONSEC. FROM 1) CODED 181-189
C
C      LTOG1 = 1    =>     JUST RECOGNIZED HIGHEST DERIVATIVE OF X.
C                          NEXT INPUT SYMBOL NOT YET RECOGNIZED
C      LTOG2 = 1    =>     JUST PROCESSED A ";",          OR JUST STARTED
C      LTOG3 = 1    =>     JUST PROCESSED A ";", "THEN", OR JUST STARTED
C      LTOG4 = 1    =>     THIS IS NOT A BLANK LINE
C      LTOG5 = 1    =>     HAVE RECOGNIZED THE DECIMAL POINT
C      LTOG6 = 1    =>     DECODING A VARIABLE NAME
C      LTOG7 = 1    =>     DECODING A NUMBER
C      LTOG8 = 1    =>     DECODING A DIFFERENTIAL EQUATION
C      LTOG9 = 1    =>     INTERPRETING A SKIP
C      LTOG10 = 1   =>     EVALUATING A DIFFERENTIAL EQUATION
C      LTOG11 = 1    =>    PROCESSING AN IF ARGUMENT
C
       SUBROUTINE COMP
       REAL  Z(100)
       DOUBLE PRECISION ZT(10), ZDOT(10), TT,ZLARGE
       COMMON ZT, ZDOT, TT, L, N, LFUNCT
       COMMON/COEF/A,B,C,D,E,F,G,H
       COMMON/STATUS/LSTAT
```

```
      INTEGER STACKA(200),STKAH(200),STACKB(40 ),STKBH(40 )
      INTEGER STACKC(200), PSCT, PSCB, N
      INTEGER PSAB, PSAT, PSBT,PI1
      REAL STACKW(50)
      INTEGER PSWT, PSLT
      LOGICAL STACKL(20)
      INTEGER INPUT(100)
      DATA LA,LB,LC,LD,LE,LF,LG,LH,LI,LJ,LK,LL,LM,LLN,
     *LO,LP,LQ,LR,LS,LT,LU,LV,LW,LX,LY,LZ
     */4HA   ,4HB   ,4HC   ,4HD   ,4HE   ,4HF   ,4HG   ,4HH   ,
     * 4HI   ,4HJ   ,4HK   ,4HL   ,4HM   ,4HN   ,4HO   ,4HP   ,
     * 4HQ   ,4HR   ,4HS   ,4HT   ,4HU   ,4HV   ,4HW   ,4HX   ,
     * 4HY   ,4HZ   /
      INTEGER PLUS,MINUS,DIV,MULT,LPAREN,RPAREN,APOS,BLANK,EQL
     *,SCOLON, PERIOD
      DATA PLUS,MINUS,DIV,MULT,LPAREN,RPAREN,APOS,BLANK,EQL
     *,SCOLON, PERIOD
     */4H+   ,4H-   ,4H/   ,4H*   ,4H(   ,4H)   ,4H'   ,4H    ,4H=   ,
     * 4H;   ,4H.   /
      INTEGER N0,N1,N2,N3,N4,N5,N6,N7,N8,N9
      DATA    N0,N1,N2,N3,N4,N5,N6,N7,N8,N9
     */4H0   ,4H1   ,4H2   ,4H3   ,4H4   ,4H5   ,4H6   ,4H7   ,4H8   ,
     *4H9   /
      WRITE(6,1200)
 1200 FORMAT('0THE SYSTEM DESCRIPTION EQUATIONS AND/OR PROGRAM:',/)
      LWFP1 = 1
      LWFP1 = 0
      LWFP2 = 1
      LWFP2 = 0
      LWITP1 = 1
      LWITP1 = 0
      LWITP2 = 1
      LWITP2 = 0
      LWUSE = 0
      LWUSE = 1
      NAVAR = 0
      NDE = 0
      Z(10) = -1.0
      N = 10
      DO 692 I=1,200
      STKAH(I) = -1
  692 CONTINUE
      PSAT = 0
      J = 0
  600 J = J + 1
C
```

```
C  READING IN DIFFERENTIAL EQUATION
C
  901 CONTINUE
      IF(PSAT.GT.0.AND.LWFP1.EQ.1) CALL DISPLY(STACKA,PSAT)
      READ(5,1030)(INPUT(I),I=1,80)
 1030 FORMAT(80A1)
      LSTOP = 0
      LTOG1 = 0
      LTOG2 = 1
      LTOG3 = 1
      LTOG4 = 0
      LTOG5 = 0
      LTOG6 = 0
      LTOG7 = 0
      LTOG8 = 0
      LTOG9 = 0
      LTOG10 = 0
      LTOG11 = 0
      RNUM = 0.0
      NPAREN = 0
      WRITE(6,1120)(INPUT(I),I=1,80)
 1120 FORMAT(' ',80A1)
C
C  BUILDING STACK A
C
      PI1 = 0
  605 PI1 = PI1 + 1
      IF(PI1.GT.80) GO TO 177
C
C  CHECKING FOR BLANK LINE INDICATING END OF EQUATIONS
C
      IF((LTOG4.EQ.0 .AND. PI1.LT.80) .OR. ( LTOG4.EQ.1)) GO TO 607
      GO TO 201
C                                                              IF, HIGH X
  607 LTOG3 = 0
      IF(INPUT(PI1).EQ.BLANK) GO TO 605
      LTOG4 = 1
      IF(INPUT( PI1 ).NE. LI) GO TO 603
      IF(INPUT(PI1+1).NE. LF) GO TO 160
      LTOG11 = 1
      STACKA(PSAT+1) = 145
      STKAH (PSAT+1) = 11
      NPAREN = 0
      PI1 =PI1 + 1
      GO TO 152
  603 IF(INPUT(PI1).NE.LX) GO TO 101
```

```
      LTOG8 = 1
      NDE = NDE + 1
      N = 0
      DO 691 I=1,10
      IF(INPUT(PI1 + I).NE.APOS) GO TO 604
      N = N + 1
  691 CONTINUE
      WRITE(6,1050)
 1050 FORMAT(' RANK OF DIFFERENTIAL EQUATION IS TO HIGH, PLEASE CORRECT'
     *)
      GO TO 601
  606 WRITE(6,1060)
 1060 FORMAT(' THE HIGHEST DERIVATIVE OF X MUST NOT HAVE A COEFFICIENT',
     *', PLEASE CORRECT')
      GO TO 601
  604 IF(LTOG2.EQ.0) GO TO 608
      STACKA(PSAT+1) = 145
      STKAH(PSAT+1) = 10
      STACKA(PSAT+2) = 146
      STKAH (PSAT+2) = -1
      PSAT = PSAT + 2
  608 STACKA(PSAT+1) = 170 + N
      STKAH(PSAT+1) = -1
      PSAT = PSAT + 1
      IF(NDE.GT.9) GO TO 309
      STACKA(PSAT+1) = 180 + NDE
      STKAH(PSAT + 1) = -1
      PSAT = PSAT + 1
  309 STACKA(PSAT + 1) = 10
      STKAH(PSAT + 1) = -1
      STACKA(PSAT + 2) = 124
      STKAH(PSAT + 2) = 8
      STACKA(PSAT + 3) = 142
      STKAH(PSAT + 3) = 1
      PSAT = PSAT + 2
      PI1 = PI1 + I - 1
C     WRITE(6,1850) J,N
 1850 FORMAT('0  N(',I1,')=',I5)
      NPAREN = 0
      LTOG1 = 1
      GO TO 134
C
 1011 PI1 = PI1 + 1
      IF(PI1.GE.80 ) GO TO 177
  101 IF(INPUT( PI1 ).EQ.BLANK) GO TO 1011
C                                                       DERIV. OF X
```

```
      IF(INPUT(PI1).NE.LX) GO TO 103
      NX = 0
      LAST = N
      IF(LTOG8.EQ.0) LAST = 9
      DO 197 I=1,LAST
      IF(INPUT(PI1 + I).NE.APOS) GO TO 102
      NX = NX + 1
  197 CONTINUE
      WRITE(6,1070)
 1070 FORMAT(' DERIVATIVE OF X TO HIGH, PLEASE CORRECT')
      GO TO 601
  102 STACKA(PSAT + 1) = I
      STKAH(PSAT + 1) = -1
      PI1 = PI1 + I - 1
      GO TO 152
C
C  RECOGNIZING THE BINARY OPERATORS
C                                                            PLUS
  103 IF(INPUT( PI1 ).NE. PLUS) GO TO 104
      STACKA(PSAT + 1) = 121
      STKAH(PSAT + 1) = 7
      GO TO 152
C                                                            MINUS
  104 IF(INPUT( PI1 ).NE.MINUS) GO TO 105
      STACKA(PSAT + 1) = 122
      STKAH(PSAT + 1) = 7
      GO TO 152
C                                                            DIV
  105 IF(INPUT( PI1 ).NE. DIV ) GO TO 106
      STACKA(PSAT + 1) = 123
      STKAH(PSAT + 1) = 8
      GO TO 134
C                                                            POWER
  106 IF(INPUT( PI1 ).NE. MULT) GO TO 107
      IF(INPUT(PI1+1).NE. MULT) GO TO 108
      STACKA(PSAT + 1) = 125
      STKAH(PSAT + 1) = 9
      PI1 = PI1 + 1
      GO TO 152
C                                                            MULT
  108 STACKA(PSAT + 1) = 124
      STKAH(PSAT + 1) = 8
      GO TO 134
C
C  RECOGNITION OF OPERANDS AND UNARY OPERATORS
C                                                            ABS
```

```
  107 IF(INPUT( PI1 ).NE.  LA ) GO TO 109
      IF(INPUT(PI1+1).NE.  LB ) GO TO 110
      IF(INPUT(PI1+2).NE.  LS ) GO TO 160
      STACKA(PSAT + 1) = 134
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GO TO 152
C                                                   A
  110 STACKA(PSAT + 1) = 11
      STKAH(PSAT + 1) = -1
      GO TO 152
C                                                   B
  109 IF(INPUT( PI1 ).NE.  LB ) GO TO 112
      STACKA(PSAT + 1) = 12
      STKAH(PSAT + 1) = -1
      GO TO 152
C                                                   COS
  112 IF(INPUT( PI1 ).NE.  LC ) GO TO 113
      IF(INPUT(PI1+1).NE.  LO ) GO TO 114
      IF(INPUT(PI1+2).NE.  LS ) GO TO 160
      STACKA(PSAT + 1) = 132
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GO TO 152
C                                                   C
  114 STACKA(PSAT + 1) = 13
      STKAH(PSAT + 1) = -1
      GO TO 152
C                                                   D
  113 IF(INPUT( PI1 ).NE.  LD ) GO TO 115
      STACKA(PSAT + 1) = 14
      STKAH(PSAT + 1) = -1
      GO TO 152
C                                                   EXP
  115 IF(INPUT( PI1 ).NE.  LE ) GO TO 116
      IF(INPUT(PI1+1).NE.  LX ) GO TO 117
      IF(INPUT(PI1+2).NE.  LP ) GO TO 160
      STACKA(PSAT + 1) = 135
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GO TO 152
C                                                   E
  117 STACKA(PSAT + 1) = 15
      STKAH(PSAT + 1) = -1
      GO TO 152
C                                                   F
```

```
  116 IF(INPUT( PI1 ).NE.  LF ) GO TO 118
      STACKA(PSAT + 1) = 16
      STKAH(PSAT + 1) = -1
      GO TO 152
C                                                        G
  118 IF(INPUT( PI1 ).NE.  LG ) GO TO 119
      STACKA(PSAT + 1) = 17
      STKAH(PSAT + 1) = -1
      GO TO 152
C                                                        H
  119 IF(INPUT( PI1 ).NE.  LH ) GO TO 3120
      STACKA(PSAT + 1) = 18
      STKAH(PSAT + 1) = -1
      GO TO 152
C                                                        INT
 3120 IF(INPUT( PI1 ).NE. LI) GO TO 120
      IF(INPUT(PI1+1).NE. LLN) GO TO 160
      IF(INPUT(PI1+2).NE. LT) GO TO 160
      STACKA(PSAT+1) = 138
      STKAH(PSAT+1) = 10
      PI1 = PI1 + 2
      GO TO 152
C                                                        TAN
  120 IF(INPUT( PI1 ).NE.  LT ) GO TO 121
      IF(INPUT(PI1+1).NE.  LA ) GO TO 122
      IF(INPUT(PI1+2).NE.  LLN) GO TO 160
      STACKA(PSAT + 1) = 133
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GO TO 152
C                                                        THEN
  122 IF(INPUT(PI1+1).NE.  LH ) GO TO 125
      IF(INPUT(PI1+2).NE.  LE ) GO TO 160
      IF(INPUT(PI1+3).NE.  LLN) GO TO 160
      PSAT = PSAT - 1
      IF(NPAREN.NE.0) GO TO 170
      PI1 = PI1 + 3
      LTOG2 = 0
      LTOG3 = 1
      LTOG11 = 0
      GO TO 152
C                                                        T
  125 STACKA(PSAT + 1) = 19
      STKAH(PSAT + 1) = -1
      GO TO 152
C                                                        SIN
```

```
  121 IF(INPUT( PI1 ).NE.  LS ) GO TO 123
      IF(INPUT(PI1+1).NE.  LI ) GO TO 701
      IF(INPUT(PI1+2).NE.  LLN) GO TO 3121
      STACKA(PSAT + 1) = 131
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GO TO 152
C                                                        SKIP
  701 IF(INPUT(PI1+1).NE.LK) GO TO 160
      IF(INPUT(PI1+2).NE.LI) GO TO 160
      IF(INPUT(PI1+3).NE.LP) GO TO 160
      PI1 = PI1 + 3
  702 PI1 = PI1 + 1
      IF(INPUT(PI1).EQ.BLANK) GO TO 702
      LTOG9 = 1
      GO TO 180
  703 LTOG9 = 0
      STACKA(PSAT+1) = 160 + RNUM + 1.5
      STKAH(PSAT+1) = -1
      PI1 = PI1 - 1
      GO TO 152
C                                                        SIGN
 3121 IF(INPUT(PI1+2).NE. LG) GO TO 160
      IF(INPUT(PI1+3).NE.LLN) GO TO 160
      STACKA(PSAT+1) = 139
      STKAH(PSAT+1) = 10
      PI1 = PI1 + 3
      GO TO 152
C                                                        LOG
  123 IF(INPUT( PI1 ).NE.  LL ) GO TO 126
      IF(INPUT(PI1+1).NE.  LO ) GO TO 124
      IF(INPUT(PI1+2).NE.  LG ) GO TO 160
      STACKA(PSAT + 1) = 137
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GO TO 152
C                                                        N
  124 IF(INPUT(PI1+1).NE.  LLN) GO TO 160
      STACKA(PSAT + 1) = 136
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 1
      GO TO 152
C
C
  170 WRITE(6,2220)
 2220 FORMAT(' UNBALANCED PARENS IN LOGICAL EXPRESSION, PLEASE CORRECT')
```

```
      GO TO 601
C                                                                       (
  126 IF(INPUT( PI1 ).NE.LPAREN) GO TO 127
      STACKA(PSAT + 1) = 142
      STKAH(PSAT + 1) = 1
      NPAREN = NPAREN + 1
C
C  TAKING CARE OF THE UNARY OPERATORS "-", AND "+"
C
  134 I = 1
  135 IF(INPUT(PI1+I).NE.BLANK) GO TO 136
      I = I + 1
      GO TO 135
  136 IF(INPUT(PI1+I).NE.MINUS) GO TO 137
      STACKA(PSAT+2) = 10
      STKAH(PSAT + 2) = -1
      STACKA(PSAT+3) = 124
      STKAH(PSAT + 3) = 8
      PSAT = PSAT + 2
      PI1 = PI1 + I
      LTOG1 = 0
      GO TO 152
  137 IF(INPUT(PI1+I).NE.PLUS ) GO TO 138
      PI1 = PI1 + I
      LTOG1 = 0
      GO TO 152
C
C  CHECK TO SEE IF LEGAL THUS FAR
C
  138 IF(LTOG1.EQ.1) GO TO 606
      PI1 = PI1 + I - 1
      GO TO 152
C                                                                       )
  127 IF(INPUT( PI1 ).NE.RPAREN) GO TO 128
      STACKA(PSAT + 1) = 143
      STKAH(PSAT + 1) = 2
      NPAREN = NPAREN - 1
      IF(NPAREN.NE.0) GO TO 152
      IF(LTOG11.NE.1) GO TO 152
      LTOG2 = 0
      LTOG3 = 1
      LTOG11 = 0
      GO TO 152
C
C  CHECK FOR EQUAL SIGN AND TAKE APROPRIATE ACTION IF FOUND
C                                                                       =
```

```
  128 IF(INPUT(PI1).NE.EQL) GO TO 161
      IF(LTOG8.NE.C) GO TO 139
      STACKA(PSAT+1) = 148
      STKAH(PSAT+1) = 0
C
C  ACTION FOLLOWING "=" IN A NORMAL EQUATION
C
 2128 PI1 = PI1 + 1
      IF(INPUT(PI1).EQ.BLANK) GO TO 2128
      IF(INPUT(PI1).EQ.PLUS) GO TO 152
      PI1 = PI1 - 1
      IF(INPUT(PI1+1).NE.MINUS) GO TO 152
      PI1 = PI1 + 1
      STACKA(PSAT+2) = 10
      STKAH(PSAT+2) = -1
      STACKA(PSAT+3) = 124
      STKAH(PSAT+3) = 8
      PSAT = PSAT + 2
      GO TO 152
C
C  ACTION FOLLOWING "=" IN A DIFFERENTIAL EQUATION
C
  139 STACKA(PSAT + 1) = 143
      STKAH(PSAT + 1) = 2
      I = 1
  129 IF(INPUT(PI1+I).NE.BLANK) GO TO 130
      I = I + 1
      GO TO 129
  130 IF(INPUT(PI1+I).NE.MINUS) GO TO 131
      STACKA(PSAT+2) = 122
      STKAH(PSAT + 2) = 7
      PI1 = PI1 + 1
      GO TO 133
  131 IF(INPUT(PI1+I).NE.PLUS ) GO TO 132
      PI1 = PI1 + 1
  132 STACKA(PSAT+2) = 121
      STKAH(PSAT + 2) = 7
  133 PI1 = PI1 + I - 1
      PSAT = PSAT + 1
      GO TO 152
C
C
  161 IF(INPUT( PI1 ).NE.PERIOD) GO TO 175
      IF(LTOG3.EQ.1) GO TO 155
C                                                        .OR.
      IF(INPUT(PI1+1).NE.LO) GO TO 162
```

```
      IF(INPUT(PI1+2).NE.LR) GO TO 160
      IF(INPUT(PI1+3).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 151
      STKAH(PSAT + 1) = 3
      PI1 = PI1 + 3
      GO TO 134
C                                                           .AND.
  162 IF(INPUT(PI1+1).NE.LA) GO TO 163
      IF(INPUT(PI1+2).NE.LLN) GO TO 160
      IF(INPUT(PI1+3).NE.LD) GO TO 160
      IF(INPUT(PI1+4).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 152
      STKAH(PSAT + 1) = 4
      PI1 = PI1 + 4
      GO TO 134
C                                                           .NOT.
  163 IF(INPUT(PI1+1).NE.LLN) GO TO 165
      IF(INPUT(PI1+2).NE.LO) GO TO 164
      IF(INPUT(PI1+3).NE.LT) GO TO 160
      IF(INPUT(PI1+4).NE.PERIOD) GO TO 160
      IF(STACKA(PSAT).NE.52) GO TO 171
      STACKA(PSAT + 1) = 153
      STKAH(PSAT + 1) = 5
      PI1 = PI1 + 4
      GO TO 134
C
  171 WRITE(6,2250)
 2250 FORMAT(' .NOT. CAN ONLY BE USED FOLLOWING .AND., PLEASE CORRECT')
      GO TO 601
C                                                           .LE.
  164 IF(INPUT(PI1+2).NE.LE) GO TO 160
      IF(INPUT(PI1+3).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 159
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GO TO 134
C                                                           .GT.
  165 IF(INPUT(PI1+1).NE.LG) GO TO 167
      IF(INPUT(PI1+2).NE.LT) GO TO 166
      IF(INPUT(PI1+3).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 154
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GO TO 134
C                                                           .GE.
  166 IF(INPUT(PI1+2).NE.LE) GO TO 160
```

```
      IF(INPUT(PI1+3).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 155
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GO TO 134
C                                                            .LT.
  167 IF(INPUT(PI1+1).NE.LL) GO TO 169
      IF(INPUT(PI1+2).NE.LT) GO TO 168
      IF(INPUT(PI1+3).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 156
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GO TO 134
C                                                            .LE.
  168 IF(INPUT(PI1+2).NE.LE) GO TO 160
      IF(INPUT(PI1+3).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 157
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GO TO 134
C                                                            .EQ.
  169 IF(INPUT(PI1+1).NE.LE) GO TO 160
      IF(INPUT(PI1+2).NE.LQ) GO TO 160
      IF(INPUT(PI1+3).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 158
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GO TO 134
C
C  DECCDING VARIABLES USED BY USER
C                                                            V
  175 IF(INPUT(PI1).NE.LV) GO TO 180
      PI1 = PI1 + 1
      LTOG6 = 1
      GO TO 180
  501 IF(RNUM.GT.0.5 .AND. RNUM.LT.40.5) GO TO 502
      WRITE(6,5501)
 5501 FORMAT('OTHE VARIABLE NUMBER IS OUT OF THE ALLOWED RANGE,'
     *' PLEASE CORRECT')
      GO TO 601
  502 IRNUM = RNUM + 0.5
      STACKA(PSAT+1) = 60 + IRNUM
      STKAH(PSAT+1) = -1
      Z(60 + IRNUM) = IRNUM
      PI1 = PI1 - 1
      GO TO 152
```

```
C
C   RECOGNIZING NUMBERS
C                                                                   NUMBERS
  180 IF(PI1.GT.160) GO TO 160
      IF(INPUT(PI1) .NE. NO) GO TO 181
      INTEG = 0
      GO TO 191
  181 IF(INPUT(PI1) .NE. N1) GO TO 182
      INTEG = 1
      GO TO 191
  182 IF(INPUT(PI1) .NE. N2) GO TO 183
      INTEG = 2
      GO TO 191
  183 IF(INPUT(PI1) .NE. N3) GO TO 184
      INTEG = 3
      GO TO 191
  184 IF(INPUT(PI1) .NE. N4) GO TO 185
      INTEG = 4
      GO TO 191
  185 IF(INPUT(PI1) .NE. N5) GO TO 186
      INTEG = 5
      GO TO 191
  186 IF(INPUT(PI1) .NE. N6) GO TO 187
      INTEG = 6
      GO TO 191
  187 IF(INPUT(PI1) .NE. N7) GO TO 188
      INTEG = 7
      GO TO 191
  188 IF(INPUT(PI1) .NE. N8) GO TO 189
      INTEG = 8
      GO TO 191
  189 IF(INPUT(PI1) .NE. N9) GO TO 190
      INTEG = 9
      GO TO 191
  190 IF(INPUT(PI1) .NE. PERIOD) GO TO 193
      IF(LTOG6.EQ.1) GO TO 501
      NDEC = 1
      LTOG5 = 1
      PI1 = PI1 + 1
      GO TO 180
C                                                                   L OF D. PNT
  191 IF(LTOG5.EQ.1) GO TO 192
      RNUM = RNUM*10.0 + INTEG
      PI1 = PI1 + 1
      LTOG7 = 1
C     WRITE(6,3111) RNUM
```

```
 3111 FORMAT('ORNUM=',F15.5)
      GO TO 180
C                                                                        R OF D. PNT
  192 RNUM = RNUM + INTEG/(10.0**NDEC)
      NDEC = NDEC + 1
      PI1 = PI1 + 1
C     WRITE(6,3111) RNUM
      GO TO 180
  193 CONTINUE
      IF(LTOG9.EQ.1) GO TO 703
      IF(LTOG6.EQ.1) GO TO 501
      IF(LTOG7.NE.1) GO TO 176
      IF(LTOG5.NE.0) GO TO 194
      IF(STACKA( PSAT ) .NE.125) GO TO 195
      STACKA( PSAT ) = 126
      GO TO 194
  195 IF(STACKA(PSAT-1).NE.125) GO TO 194
      IF (STACKA( PSAT ).NE.122) GO TO 196
      STACKA(PSAT-1) = 127
      PSAT = PSAT - 1
      GO TO 194
  196 IF(STACKA( PSAT ).NE.121) GO TO 160
      STACKA(PSAT-1) = 126
      PSAT = PSAT - 1
  194 STACKA(PSAT+1) = 20 + NAVAR + 1
      STKAH(PSAT+1) = -1
      Z(20+ NAVAR + 1) = RNUM
      NAVAR = NAVAR +1
      PI1 = PI1 - 1
      GO TO 152
C                                                                        ;
  176 IF(INPUT(PI1).NE.SCOLON) GO TO 160
  177 LTOG8 = 0
      LTOG1 = 0
      LTOG2 = 1
      LTOG3 = 1
      LTOG4 = 0
      LTOG5 = 0
      LTOG6 = 0
      LTOG7 = 0
      LTOG9 = 0
      LTOG10 = 0
      STACKA(PSAT + 1) = 147
      STKAH(PSAT+1) = -1
      PSAT = PSAT + 1
C
```

```
  156 IF(NPAREN.EQ.0) GO TO 157
      WRITE(6,2230)
 2230 FORMAT(' UNBALANCED PARENS, PLEASE CORRECT')
      GO TO 601
  157 PI1 = PI1 + 1
      IF(PI1.LT.80 ) GO TO 761
C     WRITE(6,1141)
 1141 FORMAT(100X,' FIRST PASS OK')
      GO TO 901
  761 IF(INPUT(PI1).EQ.BLANK) GO TO 157
      PSAT = PSAT - 1
      PI1 = PI1 - 1
      GO TO 152
C
  160 WRITE(6,1140)PI1
 1140 FORMAT(' WRONG SPELLING,CODE, OR ";" MISSING, CURRENT INPUT CHARAC
     *TER IS ',I5,'  PLEASE CORRECT ')
      GO TO 601
C
C  PREPARING TO RECOGNIZE NEXT INPUT CHARACTER
C
  152 PSAT = PSAT + 1
  151 PI1 = PI1 + 1
      IF(LWFP2.EQ.1) CALL DISPLY(STACKA,PSAT)
      RNUM = 0.0
      LTOG5 = 0
      LTOG6 = 0
      LTOG7 = 0
      IF(PI1.GT.80 ) GO TO 160
      IF(LTOG3.EQ.1) GO TO 607
      GO TO 101
C
  155 WRITE(6,2210)
 2210 FORMAT(' ";", OR "WHEN" MISSING, OR USE OF LOGICAL OPERATORS ',/,
     *'        IN THE DIFFERENTIAL EQUATION, PLEASE CORRECT')
      GO TO 601
C
C  CONVERSION OF INFIX TO POLISH
C
  201 PSAB = 1
      PSCT = 0
      PSBT = 0
      WRITE(6,1801)
 1801 FORMAT('0FOLLOWING IS THE CODED EQUIVALENT OF THE INPUT STRING:')
      WRITE(6,1800)(STACKA(I),I=1,PSAT)
      WRITE(6,1802)
```

```
 1802 FORMAT('OFOLLOWING IS THE CORRESPONDING HEIARCHIES ASSIGNED:')
      WRITE(6,1800)(STKAH(I),I=1,PSAT)
 1800 FORMAT(20I6)
      WRITE(6,1803)
 1803 FORMAT('OFOLLOWING IS THE MNEMONIC DECODE OF THE NUMERICALLY',
     *' CODED INPUT STRING:')
      CALL DISPLY(STACKA,PSAT)
      WRITE(6,1870)
 1870 FORMAT('0')
      WRITE(6,1820)
 1820 FORMAT(' STARTING CONVERSION TO POLISH')
C
  202 CONTINUE
      IF(LWITP1.EQ.0)GO TO 3201
 3202 IF(PSBT.GT.0) CALL DISPLY(STACKB,PSBT)
      IF(PSCT.GT.0) CALL DISPLY(STACKC,PSCT)
 3201 IF(PSAB.GT.PSAT)  GO TO 300
      IF(STKAH(PSAB).GE.0) GO TO 203
      STACKC(PSCT+1) = STACKA(PSAB)
      PSCT = PSCT + 1
      PSAB = PSAB + 1
      IF(LWITP2.EQ.0) GO TO 204
 2111 WRITE(6,2110)
      CALL DISPLY(STACKC,PSCT)
C
 2110 FORMAT(' FOLLOWING 202')
      CALL DISPLY(STACKB,PSBT)
  204 IF(PSBT.EQ.0) GO TO 202
      IF(STKBH(PSBT).LT.STKAH(PSAB)) GO TO 202
      STACKC(PSCT+1) = STACKB(PSBT)
      PSCT = PSCT + 1
      PSBT = PSBT - 1
      IF(LWITP2.EQ.0) GO TO 204
 2121 WRITE(6,2120)
 2120 FORMAT(' FOLLOWING 204')
      CALL DISPLY(STACKB,PSBT)
      CALL DISPLY(STACKC,PSCT)
      GO TO 204
C
  203 IF(STACKA(PSAB).EQ.143) GO TO 205
      STACKB(PSBT+1) = STACKA(PSAB)
      STKBH(PSBT+1) = STKAH(PSAB)
      PSAB = PSAB + 1
      PSBT = PSBT + 1
      IF(LWITP2.EQ.0) GO TO 202
 2131 WRITE(6,2130)
```

```
 2130 FORMAT(' FOLLOWING 203')
      CALL DISPLY(STACKB,PSBT)
      CALL DISPLY(STACKC,PSCT)
      GO TO 202
C
  205 IF(STACKB(PSBT).NE.142) GO TO 211
      PSAB = PSAB + 1
      PSBT = PSBT - 1
      IF(LWITP2.EQ.0) GO TO 204
 2141 WRITE(6,2140)
 2140 FORMAT(' FOLLOWING 205')
      CALL DISPLY(STACKB,PSBT)
      CALL DISPLY(STACKC,PSCT)
      GO TO 204
  211 WRITE(6,1080)
 1080 FORMAT(' DID NOT FIND "(" WHEN EXPECTED, PLEASE CHECK INPUT STRING
     * BETWEEN YOUR PARENS ')
      GO TO 601
C
C
C  PRINTING RESULTS OF COMPILATION PHASE
C
  300 WRITE(6,1081)
 1081 FORMAT('0FINISHED COMPILATION, THE DECODED POLISH STACK IS:')
      CALL DISPLY(STACKC,PSCT)
      WRITE(6,1831)
 1831 FORMAT('0 THE COMPUTER ASSIGNED VARIABLES ARE:')
      DO 393 I = 1,NAVAR
      WRITE(6,2231)I, Z(20 + I)
 2231 FORMAT(' C(',I2,') = ',F15.5)
  393 CONTINUE
      LSTAT = 0
      RETURN 1
C
      ENTRY ZDVAL
C
C  READING IN THE TEST VALUES
C
      DO 3191 I = 1,9
      Z(I) = ZT(I)
 3191 CONTINUE
      T = TT
      Z(11) = A
      Z(12) = B
      Z(13) = C
      Z(14) = D
```

```
      Z(15) = E
      Z(16) = F
      Z(17) = G
      Z(18) = H
      Z(19) = T
      LSTOP = 0
      LFUNCT = 0
      LTOG10 = 0
C
C  USING THE POLISH
C
  301 CONTINUE
C     WRITE(6,1830)
 1830 FORMAT(' USING THE POLISH')
  306 PSCB = 0
      PSWT = 0
      PSLT = 0
      LSTOP = LSTOP + 1
  302 PSCB = PSCB + 1
      IF(LSTOP.GE.2) LWUSE = 0
      SKIPNO = 1
      IF(LWUSE.EQ.1)WRITE(6,1887)
 1887 FORMAT(' ')
      IF(PSLT.GE.1.AND.LWUSE.EQ.1)WRITE(6,1881)(STACKL(I),I=1,PSLT)
 1881 FORMAT(10L2)
      IF(PSWT.GE.1.AND.LWUSE.EQ.1)WRITE(6,1880)(STACKW(I),I=1,PSWT)
 1880 FORMAT('+',20X,(1P6E20.5))
      IF(PSCB.GT.PSCT) GO TO 390
  303 NUMB = STACKC(PSCB)
      IF(NUMB.LE.0) GO TO 390
      IF(NUMB .EQ. 199) GO TO 331
      IF(NUMB.LT.100) GO TO 1
      NUMB = NUMB - 120
      IF(NUMB.LE.0) GO TO 390
  304 GO TO (21,22,23,24,25,26,26),NUMB
      NUMB = NUMB - 10
      IF(NUMB.LE.0) GO TO 390
      GO TO (31,32,33,34,35,36,37,38,39),NUMB
      NUMB = NUMB - 10
      IF(NUMB.LE.0) GO TO 390
      GO TO (390,390,390,390,45,46,47,48),NUMB
      NUMB = NUMB - 10
      IF(NUMB.LE.0) GO TO 390
      GO TO (51,52,53,54,55,56,57,58,59),NUMB
      NUMB = NUMB - 10
      IF(NUMB.LE.0) GO TO 390
```

```
      IF(NUMB.GE.10) GO TO 307
      SKIPNO = NUMB
  712 SCOUNT = 0
  711 PSCB = PSCB + 1
      IF(STACKC(PSCB).NE.147) GO TO 711
      SCOUNT = SCOUNT + 1
      IF(SCOUNT.NE.SKIPNO) GO TO 711
      GO TO 302
C
  307 NUMB = NUMB - 10
      IF(NUMB.GT.9) GO TO 308
      N = NUMB
      LTOG10 = 1
      GO TO 302
  308 NUMB = NUMB - 10
      IF(NUMB.GT.9) GO TO 390
      L = NUMB
      GO TO 302
  390 WRITE(6,1091)
 1091 FORMAT(' ILLEGAL CODE GENERATED, PLEASE CHECK INPUT STRING ')
      GO TO 601
C
C  INTERPRETING THE CODES
C
C  EVALUATING THE OPERANDS
C
    1 STACKW(PSWT+1) = Z(NUMB)
      STACKA(PSWT+1) = NUMB
      IF(NUMB.EQ.19) LFUNCT = 1
  351 PSWT = PSWT + 1
      GO TO 302
C
C  USING THE BINARY OPERATORS
C
   21 STACKW(PSWT-1) = STACKW(PSWT-1) +  STACKW(PSWT)
      GO TO 352
   22 STACKW(PSWT-1) = STACKW(PSWT-1) -  STACKW(PSWT)
      GO TO 352
   23 STACKW(PSWT-1) = STACKW(PSWT-1) /  STACKW(PSWT)
      GO TO 352
   24 STACKW(PSWT-1) = STACKW(PSWT-1) *  STACKW(PSWT)
      GO TO 352
   25 STACKW(PSWT-1) = STACKW(PSWT-1) ** STACKW(PSWT)
      GO TO 352
   26 VAL = 1.0
      LAST = STACKW(PSWT) + 0.5
```

```
      IF(LAST.EQ.0)GO TO 92
      DO 91 I=1,LAST
      VAL = VAL*STACKW(PSWT-1)
   91 CONTINUE
      STACKW(PSWT-1) = VAL
      IF(NUMB.EQ.7) STACKW(PSWT-1) = 1.0/VAL
      GO TO 352
   92 STACKW(PSWT-1) = 1.0
  352 PSWT = PSWT - 1
      GO TO 302
C
C     USING THE UNARY OPERATORS
C
   31 STACKW(PSWT) = SIN(STACKW(PSWT))
      GO TO 302
   32 STACKW(PSWT) = COS(STACKW(PSWT))
      GO TO 302
   33 STACKW(PSWT) = TAN(STACKW(PSWT))
      GO TO 302
   34 STACKW(PSWT) = ABS(STACKW(PSWT))
      GO TO 302
   35 STACKW(PSWT) = EXP(STACKW(PSWT))
      GO TO 302
   36 STACKW(PSWT) =  ALOG(STACKW(PSWT))
      GO TO 302
   37 STACKW(PSWT) = ALOG10(STACKW(PSWT))
      GO TO 302
   38 STACKW(PSWT)= INT(STACKW(PSWT))
      GO TO 302
   39 IF(STACKW(PSWT).LT.0.0)STACKW(PSWT) = -1.0
      IF(STACKW(PSWT).GE.0.0)STACKW(PSWT) =  1.0
      GO TO 302
   45 IF(.NOT. STACKL(1)) GO TO 330
      PSLT = 0
      GO TO 302
  46  STACKL(1) = .TRUE.
      GO TO 355
   47 IF(LTOG10.EQ.1) GO TO 360
      GO TO 302
   48 Z(STACKA(PSWT-1)) = STACKW(PSWT)
      PSWT = PSWT - 2
      GO TO 302
C
  330 CONTINUE
      PSLT = 0
      GO TO 712
```

```
  331 WRITE(6,2241)
 2241 FORMAT('ODIFFERENTIAL EQUATION TO USE NOT DEFINED,'
     *' PLEASE CORRECT')
      GO TO 601
C
C  USING THE LOGICAL OPERATORS
C
   51 STACKL(PSLT-1) = STACKL(PSLT-1) .OR. STACKL(PSLT)
      GO TO 354
   52 STACKL(PSLT-1) = STACKL(PSLT-1) .AND. STACKL(PSLT)
      GO TO 354
   53 STACKL(PSLT) = .NOT. STACKL(PSLT)
      GO TO 302
   54 STACKL(PSLT+1) = STACKW(PSWT-1) .GT. STACKW(PSWT)
      GO TO 353
   55 STACKL(PSLT+1) = STACKW(PSWT-1) .GE. STACKW(PSWT)
      GO TO 353
   56 STACKL(PSLT+1) = STACKW(PSWT-1) .LT. STACKW(PSWT)
      GO TO 353
   57 STACKL(PSLT+1) = STACKW(PSWT-1) .LE. STACKW(PSWT)
      GO TO 353
   58 STACKL(PSLT+1) = STACKW(PSWT-1) .EQ. STACKW(PSWT)
      GO TO 353
   59 STACKL(PSLT+1) = STACKW(PSWT-1) .NE. STACKW(PSWT)
  353 PSWT = PSWT -2
  355 PSLT = PSLT + 1
      GO TO 302
  354 PSLT = PSLT - 1
      GO TO 302
C
C  DEFINING THE DESIRED DIRIVATIVES
C
  360 DO 391 I=1,9
      IF(I.GE.N) GO TO 361
      ZDOT(I) = Z(I+1)
  391 CONTINUE
      WRITE(6,1100)N
 1100 FORMAT(' N IS TO LARGE,N=',I5)
      GO TO 601
  361 ZDOT(N) = STACKW(1)
      LAST = N
      ZLARGE = 1.0D12
      DO 392 I = 1,LAST
      IF(ZDOT(I).GT.ZLARGE) ZDOT(I) = ZLARGE
      IF(ZDOT(I).LT.-ZLARGE) ZDOT(I) = -ZLARGE
  392 CONTINUE
```

```
      IF(LWUSE.EQ.1)WRITE(6,1110)(I,Z(I),I,ZDOT(I),I=1,LAST)
 1110 FORMAT(' Z(',I1,') =',F20.5,10X,
     *       'ZDOT(',I1,') =',1PE20.5)
      LSTAT = 0
      RETURN 2
  601 CONTINUE
      LSTAT = 3
      RETURN
      END
```

```
      SUBROUTINE DISPLY (LISTI,N)
      DIMENSION LISTI(1   ), LISTO(200)
      INTEGER ANUM1(20)/
     *4HZ(1),4HZ(2),4HZ(3),4HZ(4),4HZ(5),4HZ(6),4HZ(7),4HZ(8),4HZ(9),
     *4H-1.0,4HA    ,4HB    ,4HC    ,4HD    ,4HE    ,4HF    ,4HG    ,4HH    ,
     *4HT    ,4H)    /
      INTEGER ANUM2(40)/
     *             4HC1   ,4HC2   ,4HC3   ,4HC4   ,4HC5   ,4HC6   ,4HC7   ,
     *4HC8   ,4HC9   ,4HC10  ,4HC11  ,4HC12  ,4HC13  ,4HC14  ,4HC15  ,4HC16  ,
     *4HC17  ,4HC18  ,4HC19  ,4HC20  ,4HC21  ,4HC22  ,4HC23  ,4HC24  ,4HC25  ,
     *4HC26  ,4HC27  ,4HC28  ,4HC29  ,4HC30  ,4HC31  ,4HC32  ,4HC33  ,4HC34  ,
     *4HC35  ,4HC36  ,4HC37  ,4HC38  ,4HC39  ,4HC40  /
      INTEGER ANUM3(40)/
     *4HV1   ,4HV2   ,4HV3   ,4HV4   ,4HV5   ,4HV6   ,4HV7   ,4HV8   ,4HV9   ,
     *4HV10  ,4HV11  ,4HV12  ,4HV13  ,4HV14  ,4HV15  ,4HV16  ,4HV17  ,4HV18  ,
     *4HV19  ,4HV20  ,4HV21  ,4HV22  ,4HV23  ,4HV24  ,4HV25  ,4HV26  ,4HV27  ,
     *4HV28  ,4HV29  ,4HV30  ,4HV31  ,4HV32  ,4HV33  ,4HV34  ,4HV35  ,4HV36  ,
     *4HV37  ,4HV38  ,4HV39  ,4HV40  /
      INTEGER ANUM4(40)/
     *4H+    ,4H-    ,4H/    ,4H*    ,4H**R  ,4H**PI ,4H**NI ,4HC    ,4HC    ,
     *4H0    ,4HSIN  ,4HCOS  ,4HTAN  ,4HABS  ,4HEXP  ,4HLN   ,4HLOG  ,4HINT  ,
     *4HSIGN,4H0     ,4HTHEN,4H(     ,4H)    ,4H'    ,4HIF   ,4HTRUE,4H;     ,
     *4H=    ,4H0    ,4H0    ,4HOR   ,4HAND  ,4HNOT  ,4HGT   ,4HGE   ,4HLT   ,
     *4HLE   ,4HEQ   ,4HNE   ,4H0    /
      INTEGER ANUM5(30)/
     *                               4HSK1  ,4HSK2  ,4HSK3  ,4HSK4  ,4HSK5  ,
     *4HSK6  ,4HSK7  ,4HSK8  ,4HSK9  ,4H0    ,4HN=1  ,4HN=2  ,4HN=3  ,4HN=4  ,
     *4HN=5  ,4HN=6  ,4HN=7  ,4HN=8  ,4HN=9  ,4H     ,4HL=1  ,4HL=2  ,
     *4HL=3  ,4HL=4  ,4HL=5  ,4HL=6  ,4HL=7  ,4HL=8  ,4HL=9  ,4HTIME/
      IF(N.EQ.0) GO TO 99
      DO 91 I=1,N
      NUMB=LISTI(I)
      IF(NUMB.GT.20) GO TO 11
      LISTO(I) = ANUM1(NUMB)
      GO TO 91
   11 NUMB = NUMB - 20
      IF(NUMB.GT.40) GO TO 12
      LISTO(I) = ANUM2(NUMB)
      GO TO 91
   12 NUMB = NUMB - 40
      IF(NUMB.GT.40) GO TO 14
      LISTO(I) = ANUM3(NUMB)
      GO TO 91
   14 NUMB = NUMB - 60
      IF(NUMB.GT.40) GO TO 15
      LISTO(I) = ANUM4(NUMB)
```

```
      GO TO 91
   15 NUMB = NUMB - 40
      IF(NUMB.GT.30) GO TO 13
      LISTO(I) = ANUM5(NUMB)
      GO TO 91
   13 WRITE(6,140)LISTI(I)
  140 FORMAT('C CODE NO. IS OUT OF ALLOWED RANGE, IT IS:',I5)
      RETURN
   91 CONTINUE
      WRITE(6,100)(LISTO(I),I=1,N)
  100 FORMAT('0', 20A6)
      WRITE(6,110)
  110 FORMAT('0')
      RETURN
   99 WRITE(6,120)N
  120 FORMAT(' N=',I5)
      RETURN
      END
```

```
      SUBROUTINE VPLOT(XY,JXY,N,NDIM,NCUR,ISCALE,XL,XU,YL,YU)
      DIMENSION IGRID(101),XS(11),YS(17),ICHAR(7),XY(1),JXY(1)
      DATA ICHAR/1H+,1H*,1H¬,1H$,1H=,1H.,1H /
      XMIN=XL
      XMAX=XU
      YMIN=YL
      YMAX=YU
      IF(ISCALE.NE.0)GO TO 32
      XMAX=-1.0E 20
      XMIN=-XMAX
      YMAX=XMAX
      YMIN=XMIN
      J2=0
      DO 31 J=1,NCUR
      J2=J2+2
      JIX=(JXY(J2-1)-1)*NDIM
      JIY=(JXY(J2)-1)*NDIM
      DO 31 I=1,N
      IJX=JIX+I
      IJY=JIY+I
      IF(XY(IJX).GT.XMAX)XMAX=XY(IJX)
      IF(XY(IJX).LT.XMIN)XMIN=XY(IJX)
      IF(XY(IJY).GT.YMAX)YMAX=XY(IJY)
      IF(XY(IJY).LT.YMIN)YMIN=XY(IJY)
   31 CONTINUE
   32 XR=XMAX-XMIN
      IF(XR.EQ.0.0)XR=1.0E-20
      YR=YMAX-YMIN
      IF(YR.EQ.0.0)YR=1.0E-20
      XT=XMAX*XMIN
      YT=YMIN*YMAX
      IF(XT.LT.0.0)IYAX=100.0*(-XMIN)/XR+1.5
      IF(YT.LE.0.0)IXAX=64.0*YMAX/YR+1.5
      XINCR=XR/10.0
      YINCR=YR/16.0
      XS(1)=XMIN
      YS(1)=YMAX
      DO 46 I=2,11
   46 XS(I)=XS(I-1)+XINCR
      DO 47 I=2,17
   47 YS(I)=YS(I-1)-YINCR
      WRITE(6,10)(XS(I),I=1,11)
      II=1
      KK=0
      DO 146 LINE=1,65
      DO 101 J=1,101
```

```
101  IGRID(J)=ICHAR(7)
     IF(YT.GT.0.0)GO TO 109
     IF(LINE.NE.IXAX)GO TO 109
     DO 105 J=1,101
105  IGRID(J)=ICHAR(6)
109  IF(XT.LT.0.0)IGRID(IYAX)=ICHAR(6)
     J2=0
     DO 125 J=1,NCUR
     JC=MOD(J,5)
     J2=J2+2
     JIX=(JXY(J2-1)-1)*NDIM
     JIY=(JXY(J2)-1)*NDIM
     DO 125 I=1,N
     IJX=JIX+I
     IJY=JIY+I
     IPTY=64.0*(YMAX-XY(IJY))/YR+1.5
     IF(IPTY.GT.65)IPTY=65
     IF(IPTY.LT.1)IPTY=1
     IF(IPTY.NE.LINE)GO TO 125
     IPTX=100.0*(XY(IJX)-XMIN)/XR+1.5
     IF(IPTX.LT.1)IPTX=1
     IF(IPTX.GT.101)IPTX=101
     IF(JC.NE.0)GO TO 119
     IGRID(IPTX)=ICHAR(5)
     GO TO 125
119  IGRID(IPTX)=ICHAR(JC)
125  CONTINUE
     IF(KK.GT.0)GO TO 134
     WRITE(6,20)YS(II),(IGRID(I),I=1,101),YS(II)
     II=II+1
     GO TO 135
134  WRITE(6,30)(IGRID(I),I=1,101)
135  KK=KK+1
     IF(KK.NE.4)GO TO 146
     KK=0
146  CONTINUE
     WRITE(6,40)(XS(I),I=1,11)
     WRITE(6,50)XMAX,XMIN,YMAX,YMIN
 50  FORMAT('0MAXES ARE',4F15.4)
     RETURN
 10  FORMAT(1H1,1PE15.2,10E10.2/10X,1H*,20(5H+****),2H+*)
 20  FORMAT(1PE10.2,1H+,101A1,1H+,E9.2)
 30  FORMAT(10X,1H*,101A1,1H*)
 40  FORMAT(10X,1H*,20(5H+****),2H+*/1PE16.2,10E10.2)
     END
```

PROGRAM B

LISTING OF INTERACTIVE GRAPHICS VERSION

```
      DOUBLE PRECISION Z(10), ZDOT(10), DEL, T, ITITLE(12)
      DIMENSION ZI(9),XZ(50),YZ(50)
      INTEGER ITDIR(41),IGDIR(5),ITEXT(24,40),IMAGE(4812)
      INTEGER STACKC(200),IMSOLV(402)
      DIMENSION ZC(100)
      COMMON ZC,Z,ZDOT,DEL,T,A,B,C,D,E,F,G,H,
     *XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,TI,TF,ZI,
     *XMIN,XMAX,YMIN,YMAX,XFACT,YFACT
      COMMON IDEV,NO,NN,LSTAT,N,STACKC,LFUNCT,L
      COMMON XCENTA, YCENTA
      OUTPUT(3)'MAIN'
C
C  THIS IS THE MAIN PROGRAM
C
      M=0
      LSTAT = 0
      T = 0.0
      CALL INIT(ITDIR,IGDIR,ITEXT,IMAGE)
   11 CALL SETUP2(ITDIR,IGDIR,ITEXT,IMAGE,K,K2,XZ,YZ)
      IF(M.EQ.0) GO TO 12
      GO TO (11,12,13,14),K
   12 CALL COMP(ITEXT)
      M = 1
      IF(LSTAT.NE.0) GO TO 11
   13 CALL GRID(ITDIR,IGDIR,ITEXT,IMAGE)
      CALL SLOPES(ITDIR,IGDIR,ITEXT,IMAGE)
      OUTPUT(3)K2,XZ,YZ
   14 DO 197 MM=1,K2
      ZI(1)=XZ(MM)
      ZI(2)=YZ(MM)
      ENCODE(96,10,ITEXT(1,1))TI,DEL,TF,(ZI(J),J=1,8)
   10 FORMAT('INITIAL TIME =',F10.5,'  TIME STEP =',F10.5,
     *'  FINAL TIME =',F10.5,/,
     *'IC(1)=',F11.5,'   IC(2)=',F11.5,'   IC(3)=',F11.5,'   IC(4)=',
     *F11.5,'   EDIT-TO-GO LINE',/,
```

```
     *'IC(5)=',F11.5,'    IC(6)=',F11.5,'    IC(7)=',F11.5,'    IC(8)=',
     *F11.5)
      DO 91 I = 1,2
      CALL TEXTO(IDEV,ITEXT(1,I),24,I,1,1,2,IER)
 91   CONTINUE
      CALL SOLVE(ITDIR,IGDIR,ITEXT,IMAGE,IMSOLV)
      IF(LSTAT.NE.0) GO TO 11
 197  CONTINUE
C
C  ERASING THE OLD STARTING VALUES
C
      IMAGE(1) = IHEAD(0,7)
      DO 194 I = 2,30
 194  IMAGE(I) = IPACK(0.0,0.0,0)
      CALL GRAPHO(IDEV,IMAGE,30,1,IER)
      GO TO 11
      END
```

```
      SUBROUTINE INIT(ITDIR,IGDIR,ITEXT,IMAGE)
      DOUBLE PRECISION Z(10), ZDOT(10), DEL, T, ITITLE(12)
      DIMENSION ZI(9), ITEMP(24)
      INTEGER ITDIR(41),IGDIR(5),ITEXT(24,40),IMAGE(4812)
      INTEGER STACKC(200)
      DIMENSION ZC(100)
      COMMON ZC,Z,ZDOT,DEL,T,A,B,C,D,E,F,G,H,
     *XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,TI,TF,ZI,
     *XMIN,XMAX,YMIN,YMAX,XFACT,YFACT
      COMMON IDEV,NO,NN,LSTAT,N,STACKC,LFUNCT,L
      NAME LIST
      OUTPUT(3)' INTO INIT '
      OUTPUT(101)' ENTER IDEV=1 OR 2'
      INPUT(101)
      CALL DTINIT(IDEV,ITDIR,41,IER)
      CALL DGINIT(IDEV,IGDIR,5,IER)
      IMAGE(1)=IHEAD(0,7)
      DO 91 I=2,30
   91 IMAGE(I)=IPACK(0.0,0.0,0)
      CALL GRAPHO(IDEV,IMAGE,30,1,IER)
      DO 95 I=1,40
      DO 95 J=1,24
   95 ITEXT(J,I) = 77777777B
      J=0
C
C  SENDING OUT TEXT IN THE ORDER BEST FOR FUTURE EDITING AT THE TERMINAL
C
      DO 93 I=1,2
      IF(IER.NE.0)CALL ERROR(4HSET1,4H 71 ,I,J)
   71 CALL TEXTO(IDEV,ITEXT(1,I),24,I,1,1,2,IER)
   93 CONTINUE
      DO 94 I=37,40
   72 CALL TEXTO(IDEV,ITEXT(1,I),24,I,1,1,2,IER)
      IF(IER.NE.0)CALL ERROR(4HSET1,4H 72 ,I,J)
   94 CONTINUE
```

```
      D0 96 I=3,36
   74 CALL TEXT0(IDEV,ITEXT(1,I),24,I,1,1,2,IER)
      IF(IER.NE.0)CALL ERR0R(4HSET1,4H 74 ,I,J)
   96 C0NTINUE
C
C  DEFINING SAMPLE PR0BLEM T0 BE USED F0R DEBUGGING AND DEM0NSTRATI0N
C
      ENC0DE(96,10,ITEXT(1,5))
   10 F0RMAT($ X'' - (1.0 - X**2)*X' + B*X = 0.0$)
      XSCALE = 1.0
      YSCALE = 1.0
      XCENT = 0.0
      YCENT = 0.0
      NXSIZE = 8
      NYSIZE = 8
      A = 1.0
      B = 1.0
      C = 0.0
      D = 0.0
      E = 0.0
      F = 0.0
      G = 0.0
      H = 0.0
      TI = 0.0
      DEL = 0.1
      TF = 30.0
      ZI(1) = 0.05
      D0 97 I = 2,8
      ZI(I) = 0.0
   97 C0NTINUE
      0UTPUT(3)' LEAVING INIT'
      0UTPUT(3)A,B,C,D,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
     *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,N0,NN,LSTAT,N
      RETURN
      END
```

```
C  THE FØLLØWING SUBRØUTINE IS CALLED IF AN ERRØR ØCCURES WHEN
C  CALLING TEXTØ ØR GRAPHØ.
C
      SUBRØUTINE ERRØR(ISUB,LØC,I,J)
      WRITE(6,10)ISUB,LØC
   10 FØRMAT(2A10)
C     ØUTPUT(6)I,J
      RETURN
      END
```

```
C
C     THIS SUBROUTINE IS THE CENTER FOR THE INTERACTION DURING THE
C     RUNNING OF THIS PROGRAM.  IT SENDS OUT THE TEXT INFORMATION
C     NEEDED FOR THE INTERACTION OF THE PROGRAM AND WAITS WHILE THE
C     DESIRED CHANGES ARE BEING MADE.  ON THE BASIS OF THE CHANGES
C     MADE IT WILL PROCEED ON TO GET A SOLUTION OR GO BACK
C     AND RECOMPILE THE SYSTEM EQUATIONS AND/OR CALL
C     SUBROUTINE SLOPES AND GET A NEW SET OF SLOPES FIRST AND THEN
C     PROCEED ON TO GET A SOLUTION.  AFTER THE SOLUTION IS OBTAINED
C     THE PROGRAM RETURNS HERE TO COMPLETE THE LOOP.  THIS INTERACTION
C     CONTINUES UNTIL AN INITIAL TIME OF -99.9 IS INSERTED AT
C     WHICH TIME THE PROGRAM STOPS.
C
      SUBROUTINE SETUP2(ITDIR,IGDIR,ITEXT,IMAGE,K,K2,XZ,YZ)
      DIMENSION XZ(50), YZ(50)
      DOUBLE PRECISION Z(10), ZDOT(10), DEL, T, ITITLE(12)
      DIMENSION ZI(9)
      INTEGER ITDIR(41),IGDIR(5),ITEXT(24,40),IMAGE(4812)
      INTEGER STACKC(200)
      DIMENSION ZC(100)
      DIMENSION IBLANK(24), ITEMP1(24), ITEMP2(24)
      COMMON ZC,Z,ZDOT,DEL,T,A,B,C,D,E,F,G,H,
     *XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,TI,TF,ZI,
     *XMIN,XMAX,YMIN,YMAX,XFACT,YFACT
      COMMON IDEV,NO,NN,LSTAT,N,STACKC,LFUNCT,L
      NAME LIST
   24 CONTINUE
      OUTPUT(3)' INTO SETUP2 '
      OUTPUT(3)A,B,C,D,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
     *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,NO,NN,LSTAT,N
      K = 4
      IF(LSTAT.EQ.2)ENCODE(96,130,ITEXT(1,28))
  130 FORMAT(' COMPILE ERROR,  PLEASE RECHECK SYSTEM DESCRIPTION AND TRY
     * AGAIN')
      IF(LSTAT.EQ.3)ENCODE(96,140,ITEXT(1,28))
```

```
 140 FORMAT(' EXECUTION ERROR, PLEASE RECHECK SYSTEM DESCRIPTION AND TR
    *Y AGAIN')
     CALL TEXTO(IDEV,ITEXT(1,28),24,28,1,1,2,IER)
     DO 291 I = 1,24
     IBLANK(I) = 77777777B
 291 CONTINUE
     ENCODE(96,232,ITEMP1)
 232 FORMAT(' DO YOU DESIRE TO START A COMPLETELY NEW PROBLEM, ANSWER Y
    *ES OR NO')
     CALL TEXTO(IDEV,ITEMP1,24,30,1,1,2,IER)
     CALL TEXTR(IDEV,IBLANK,24,32,1,1,2,IER)
 395 IF(MOD(ITDIR(36),8).EQ.0) GO TO 395
     CALL TEXTO(IDEV,IBLANK,24,28,1,1,2,IER)
     DO 493 J = 1,24
     ITEXT(J,28) = 77777777B
493  CONTINUE
     CALL TEXTI(IDEV,ITEMP1,24,32,1,IER)
C  SKIP AROUND IF ANSWER IS NOT YES
     CALL TEXTO(IDEV,IBLANK,24,32,1,1,2,IER)
     IF(ITEMP1(1).NE.70256260B) GO TO 227
     IMAGE(1) = IHEAD(0,7)
     DO 491 I = 2,1060
 491 IMAGE(I) = IPACK(0.0,0.0,0)
     CALL GRAPHO(IDEV,IMAGE,400,4,IER)
     CALL GRAPHO(IDEV,IMAGE,1060,3,IER)
     CALL GRAPHO(IDEV,IMAGE,6,2,IER)
     K = 2
     I = 0
     LT = 4
 200 I = I + 1
     GO TO (201,202,203,204,205,206,207,208,209,210,
    *211,212,213,214,215,216,217,218,219,220,
    *221,222,223,224,225,226,227),I
 201 ENCODE(96,301,ITEMP1)
 301 FORMAT(' TYPE IN SYSTEM DESCRIPTION,  A NULL LINE ENDS THIS PHASE
```

```
     *')
      CALL TEXT0(IDEV,ITEMP1,24,30,1,1,2,IER)
  401 LT = LT + 1
      CALL TEXTR(IDEV,IBLANK,24,LT,1,1,2,IER)
      LTT = LT + 4
  392 IF(MOD(ITDIR(LTT),8).EQ.0) GO TO 392
      CALL TEXTI(IDEV,ITEXT(1,LT),24,LT,1,IER)
      DO 393 L = 1,24
      IF(ITEXT(L,LT).NE.606060603) GO TO 401
  393 CONTINUE
      GO TO 200
  202 ENCODE(96,302,ITEMP1)
  302 FORMAT(' TYPE IN VALUE FOR X-SCALE, (UNITS/INCH, F10.0 FORMAT)')
      GO TO 231
  203 ENCODE(96,303,ITEMP1)
  303 FORMAT(' TYPE IN VALUE FOR X-CENTER, (VALUE IN F10.0 FORMAT)')
      GO TO 231
  204 ENCODE(96,304,ITEMP1)
  304 FORMAT(' TYPE IN VALUE FOR X-SIZE, (INCHES IN F10.0 FORMAT)')
      GO TO 231
  205 ENCODE(96,305,ITEMP1)
  305 FORMAT(' TYPE IN VALUE FOR Y-SCALE, (UNITS/INCH, F10.0 FORMAT)')
      GO TO 231
  206 ENCODE(96,306,ITEMP1)
  306 FORMAT(' TYPE IN VALUE FOR Y-CENTER, (VALUE IN F10.0 FORMAT)')
      GO TO 231
  207 ENCODE(96,307,ITEMP1)
  307 FORMAT(' TYPE IN VALUE FOR Y-SIZE, (INCHES IN F10.0 FORMAT)')
      GO TO 231
  208 ENCODE(96,308,ITEMP1)
  308 FORMAT(' TYPE IN VALUE FOR A, ( F10.0 FORMAT)')
      GO TO 231
  209 ENCODE(96,309,ITEMP1)
  309 FORMAT(' TYPE IN VALUE FOR B, ( F10.0 FORMAT)')
      GO TO 231
```

```
210 ENCØDE(96,310,ITEMP1)
310 FØRMAT(' TYPE IN VALUE FØR C, ( F10.0 FØRMAT)')
    GØ TØ 231
211 ENCØDE(96,311,ITEMP1)
311 FØRMAT(' TYPE IN VALUE FØR D, ( F10.0 FØRMAT)')
    GØ TØ 231
212 ENCØDE(96,312,ITEMP1)
312 FØRMAT(' TYPE IN VALUE FØR E, ( F10.0 FØRMAT)')
    GØ TØ 231
213 ENCØDE(96,313,ITEMP1)
313 FØRMAT(' TYPE IN VALUE FØR F, ( F10.0 FØRMAT)')
    GØ TØ 231
214 ENCØDE(96,314,ITEMP1)
314 FØRMAT(' TYPE IN VALUE FØR G, ( F10.0 FØRMAT)')
    GØ TØ 231
215 ENCØDE(96,315,ITEMP1)
315 FØRMAT(' TYPE IN VALUE FØR H, ( F10.0 FØRMAT)')
    GØ TØ 231
216 ENCØDE(96,316,ITEMP1)
316 FØRMAT(' TYPE IN VALUE FØR INITIAL TIME ( F10.0 FØRMAT )')
    GØ TØ 231
217 ENCØDE(96,317,ITEMP1)
317 FØRMAT(' TYPE IN VALUE FØR TIME STEP ( F10.0 FØRMAT )')
    GØ TØ 231
218 ENCØDE(96,318,ITEMP1)
318 FØRMAT(' TYPE IN VALUE FØR FINAL TIME ( F10.0 FØRMAT )')
    GØ TØ 231
219 ENCØDE(96,319,ITEMP1)
319 FØRMAT(' TYPE IN VALUE FØR FIRST I.C., X(1) USING F10.0 FØRMAT')
    GØ TØ 231
220 ENCØDE(96,320,ITEMP1)
320 FØRMAT(' TYPE IN VALUE FØR SECØND I.C., X(2) USING F10.0 FØRMAT')
    GØ TØ 231
221 ENCØDE(96,321,ITEMP1)
321 FØRMAT(' TYPE IN VALUE FØR THIRD I.C., X(3) USING F10.0 FØRMAT')
```

```
      GØ TØ 231
  222 ENCØDE(96,322,ITEMP1)
  322 FØRMAT(' TYPE IN VALUE FØR FØURTH I.C., X(4) USING F10.0 FØRMAT')
      GØ TØ 231
  223 ENCØDE(96,323,ITEMP1)
  323 FØRMAT(' TYPE IN VALUE FØR FIFTH I.C., X(5) USING F10.0 FØRMAT')
      GØ TØ 231
  224 ENCØDE(96,324,ITEMP1)
  324 FØRMAT(' TYPE IN VALUE FØR SIXTH I.C., X(6) USING F10.0 FØRMAT')
      GØ TØ 231
  225 ENCØDE(96,325,ITEMP1)
  325 FØRMAT(' TYPE IN VALUE FØR SEVENTH I.C., X(7) USING F10.0 FØRMAT')
      GØ TØ 231
  226 ENCØDE(96,326,ITEMP1)
  326 FØRMAT(' TYPE IN VALUE FØR EIGHTH I.C., X(8) USING F10.0 FØRMAT')
C
  231 CALL TEXTØ(IDEV,ITEMP1,24,30,1,1,2,IER)
      CALL TEXTR(IDEV,IBLANK,24,32,1,1,2,IER)
  391 IF(MØD(ITDIR(36),8).EQ.0) GØ TØ 391
      CALL TEXTI(IDEV,ITEMP1,24,32,1,IER)
      GØ TØ (200,252,253,254,255,256,257,258,259,260,
     *261,262,263,264,265,266,267,268,269,270,
     *271,272,273,274,275,276),I
C
  252 DECØDE(96,352,ITEMP1)XSCALE
  352 FØRMAT(F20.0)
      GØ TØ 281
  253 DECØDE(96,352,ITEMP1)XCENT
      GØ TØ 281
  254 DECØDE(96,352,ITEMP1)XSIZE
      NXSIZE = XSIZE + 0.5
      GØ TØ 281
  255 DECØDE(96,352,ITEMP1)YSCALE
      GØ TØ 281
  256 DECØDE(96,352,ITEMP1)YCENT
```

```
      GØ TØ 281
  257 DECØDE(96,352,ITEMP1)YSIZE
      NYSIZE = YSIZE + 0.5
  281 ENCØDE(96,20,ITEXT(1,37))XSCALE,XCENT,XSIZE,YSCALE,YCENT,YSIZE
      DØ 292 J = 37,38
      CALL TEXTØ(IDEV,ITEXT(1,J),24,J,1,1,2,IER)
  292 CØNTINUE
      GØ TØ 200
  258 DECØDE(96,352,ITEMP1)A
      GØ TØ 282
  259 DECØDE(96,352,ITEMP1)B
      GØ TØ 282
  260 DECØDE(96,352,ITEMP1)C
      GØ TØ 282
  261 DECØDE(96,352,ITEMP1)D
      GØ TØ 282
  262 DECØDE(96,352,ITEMP1)E
      GØ TØ 282
  263 DECØDE(96,352,ITEMP1)F
      GØ TØ 282
  264 DECØDE(96,352,ITEMP1)G
      GØ TØ 282
  265 DECØDE(96,352,ITEMP1)H
C
  282 ENCØDE(96,30,ITEXT(1,39))A,B,C,D,E,F,G,H
      DØ 293 J = 39,40
      CALL TEXTØ(IDEV,ITEXT(1,J),24,J,1,1,2,IER)
  293 CØNTINUE
      GØ TØ 200
  266 DECØDE(96,352,ITEMP1)TI
      GØ TØ 283
  267 DECØDE(96,352,ITEMP1)DEL
      GØ TØ 283
  268 DECØDE(96,352,ITEMP1)TF
      GØ TØ 283
```

```
  269 DECODE(96,352,ITEMP1)ZI(1)
      GO TO 283
  270 DECODE(96,352,ITEMP1)ZI(2)
      GO TO 283
  271 DECODE(96,352,ITEMP1)ZI(3)
      GO TO 283
  272 DECODE(96,352,ITEMP1)ZI(4)
      GO TO 283
  273 DECODE(96,352,ITEMP1)ZI(5)
      GO TO 283
  274 DECODE(96,352,ITEMP1)ZI(6)
      GO TO 283
  275 DECODE(96,352,ITEMP1)ZI(7)
      GO TO 283
  276 DECODE(96,352,ITEMP1)ZI(8)
C
  283 ENCODE(96,10,ITEXT(1,1))TI,DEL,TF,(ZI(J),J=1,8)
      DO 294 J=1,3
      CALL TEXTO(IDEV,ITEXT(1,J),24,J,1,1,2,IER)
  294 CONTINUE
      GO TO 200
  227 CONTINUE
      ENCODE(96,150,ITEXT(1,30))
  150 FORMAT('MAKE DESIRED CHANGES AND CHOOSE NEW INITIAL CONDITIONS','
     *(TEXT OR LIGHT PEN) TO INITIATE NEW RUN')
      DO 91 I=1,40
      IF(I.GE.4.AND.I.LE.31) GO TO 91
      DO 91 J=1,24
      ITEXT(J,I)=77777777B
   91 CONTINUE
      ENCODE(96,10,ITEXT(1,1))TI,DEL,TF,(ZI(J),J=1,8)
   10 FORMAT('INITIAL TIME =',F10.5,'  TIME STEP =',F10.5,
     *'  FINAL TIME =',F10.5,/,
     *'IC(1)=',F11.5,'   IC(2)=',F11.5,'   IC(3)=',F11.5,'   IC(4)=',
     *F11.5,'   EDIT-TO-GO LINE',/,
```

```
     *'IC(5)=',F11.5,'    IC(6)=',F11.5,'    IC(7)=',F11.5,'    IC(8)=',
     *F11.5)
      XSIZE = NXSIZE
      YSIZE = NYSIZE
      ENCODE(96,20,ITEXT(1,37))XSCALE,XCENT,XSIZE,YSCALE,YCENT,YSIZE
   20 FORMAT('XSCALE =',F10.5,'  XCENTER =',F10.5,'  XSIZE =',F10.5,/,
     *       'YSCALE =',F10.5,'  YCENTER =',F10.5,'  YSIZE =',
     *F10.5)
      ENCODE(96,30,ITEXT(1,39))A,B,C,D,E,F,G,H
   30 FORMAT('A=',F18.5,'  B=',F18.5,'  C=',F18.5,'  D=',F18.5,//,
     *       'E=',F18.5,'  F=',F18.5,'  G=',F18.5,'  H=',F18.5)
      DO 195 I = 5,27
      DO 195 J = 21,24
      ITEXT(J,I) = 77777777B
  195 CONTINUE
      WRITE(3,910)(((ITEXT(I,J)),I=1,24),J=1,40)
  910 FORMAT(24A4)
      DO 94 I=1,40
      CALL TEXTO(IDEV,ITEXT(1,I),24,I,1,1,2,IER)
   94 CONTINUE
   22 CONTINUE
      K2=2
      IF(MOD(ITDIR(2),8).NE.0)GO TO 23
      IF(MOD(IGDIR(1),8).NE.0)K1=3; GO TO 23
      GO TO 22
   23 DO 93 I=3,6
      IF(MOD(ITDIR(I),8).NE.0.AND.K.NE.2) K = 3
   93 CONTINUE
      DO 96 I=8,40
      IF(MOD(ITDIR(I),8).NE.0)K=2
   96 CONTINUE
      DO 492 I = 2,1060
  492 IMAGE(I) = IPACK(0.0,0.0,0)
      CALL GRAPHO(IDEV,IMAGE,400,4,IER)
      IF(K.NE.2.AND.K.NE.3) GO TO 489
```

```
      CALL GRAPHO(IDEV,IMAGE,1060,3,IER)
      CALL GRAPHO(IDEV,IMAGE,6,2,IER)
  489 CONTINUE
      IGDIR(1) = LAND(IGDIR(1),77777770B)
      DO 95 I=1,40
      CALL TEXTI(IDEV,ITEXT(1,I),24,I,1,IER)
   95 CONTINUE
      DO 194 J = 1,24
      ITEXT(J,30) = 77777777B
  194 CONTINUE
      CALL TEXTO(IDEV,ITEXT(1,30),24,30,1,1,2,IER)
      DECODE(96,10,ITEXT(1,1))TI,DEL,TF,(ZI(J),J=1,8)
      XZ(1)=ZI(1)
      YZ(1)=ZI(2)
      IF(K1.NE.3)GO TO 31
      CALL GRAPHI(IDEV,IMAGE,1,IER)
      K2=0
      DO 192 I=2,20
      CALL UNPACK(IMAGE(I),XN,YN,IZ)
      IF(IZ.EQ.0) GO TO 192
      K2=K2 + 1
      CALL UNORM(XZ(K2),YZ(K2),XN,YN)
  192 CONTINUE
      K1=0
   31 DECODE(96,20,ITEXT(1,37))XSCALE,XCENT,XSIZE,YSCALE,YCENT,YSIZE
      DECODE(96,30,ITEXT(1,39))A,B,C,D,E,F,G,H
      CALL SNORM(XSCALE)
      CALL SNORM(YSCALE)
      IF(XSIZE.GT.8.0)XSIZE = 8.0
      IF(YSIZE.GT.8.0)YSIZE = 8.0
      NXSIZE = XSIZE + 0.5
      NYSIZE = YSIZE + 0.5
      IF(ZI(1).EQ.-99.9) STOP
   21 CONTINUE
      K2=K2-1
```

```
      ENCØDE(96,10,ITEXT(1,1))TI,DEL,TF,(ZI(J),J=1,8)
      ENCØDE(96,20,ITEXT(1,37))XSCALE,XCENT,XSIZE,YSCALE,YCENT,YSIZE
      ENCØDE(96,30,ITEXT(1,39)) A,B,C,D,E,F,G,H
      DØ 193 I = 1,40
      CALL TEXTØ(IDEV,ITEXT(1,I),24,I,1,1,2,IER)
  193 CØNTINUE
      ØUTPUT(3)'LEAVING SETUP2'
      ØUTPUT(3)A,B,C,D,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
     *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,N9,NN,LSTAT,N
      ØUTPUT(3) DEL,T
      WRITE(3,8110)ZC,Z,ZDØT
      WRITE(3,8111)STACKC
 8110 FØRMAT(5X,10F12.3)
 8111 FØRMAT(5X,10I12)
      RETURN
      END
```

```
C
      SUBRØUTINE UNØRM(X,Y,XN,YN)
      DØUBLE PRECISIØN Z(10), ZDØT(10), DEL, T
      DIMENSIØN ZI(9)
      DIMENSIØN ZC(100)
      INTEGER STAC<C(200)
      CØMMØN ZC,Z,ZDØT,DEL,T,A,B,C,D,E,F,G,H,
     *XSCALE,YSCALE,XCENTA,YCENTA,NXSIZE,NYSIZE,TI,TF,ZI,
     *XMIN,XMAX,YMIN,YMAX,XFACT,YFACT
      CØMMØN IDEV,NØ,NN,LSTAT,N,STACKC,LFUNCT,L
      CØMMØN XCENT, YCENT
      X=(XN + XCENT)/XFACT
      Y=(YN + YCENT + 0.1)/YFACT
      RETURN
      END
```

```
C
C
      SUBRØUTINE SNØRM(SCALE)
      AN = 0.1
      N = 0
      IF(SCALE.GT.1.0) AN = 10.0
   21 IF(SCALE.GE.1.0.AND. SCALE.LT.10.0) GØ TØ 22
      N = N + 1
      SCALE = SCALE/AN
      GØ TØ 21
   22 NSCALE = SCALE + 0.5
      SCALE = NSCALE
      IF(N.EQ.0) RETURN
      DØ 91 I=1,N
      SCALE = SCALE*AN
   91 CØNTINUE
      RETURN
      END
```

```
C  THE FOLLOWING SUBROUTINE IS CALLED IF AN ERROR OCCURES WHEN
C  CALLING TEXTO OR GRAPHO.
C
      SUBROUTINE ERROR(ISUB,LOC,I,J)
      WRITE(6,10)ISUB,LOC
   10 FORMAT(2A10)
C     OUTPUT(6)I,J
      RETURN
      END
```

```
C
C     SUBROUTINE COMP
C
C  THE NUMERICAL CODES ASSIGNED TO THE VARIOUS OPERANDS
C  AND OPERATORS ARE LISTED BELOW
C
C    'Z(1)'    1     -1
C    'Z(2)'    2     -1
C    'Z(3)'    3     -1
C    'Z(4)'    4     -1
C    'Z(5)'    5     -1
C    'Z(6)'    6     -1
C    'Z(7)'    7     -1
C    '7(8)'    8     -1
C    'Z(9)'    9     -1
C    '-1.0'    10    -1
C    'A'       11    -1
C    'B'       12    -1
C    'C'       13    -1
C    'D'       14    -1
C    'E'       15    -1
C    'F'       16    -1
C    'G'       17    -1
C    'H'       18    -1
C    'T'       19    -1
C
C    COMPILER ASSIGNED VARIABLES ARE Z(21) THRU Z(60)
C
C    USER ASSIGNED VARIABLES ARE V1 THRU V40  CORRESPONDING
C              TO INTERNAL DESIGNATIONS Z(61) THRU Z(100)
C
C     PLUS    121    7
C     MINUS   122    7
C     DIV     123    8
C     MULT    124    8
```

```
C       PWR     125    9
C       PIPWR   126    9
C       PNPWR   127    9
C
C      'SIN'    131    10
C      'CØS'    132    10
C      'TAN'    133    10
C      'ABS'    134    10
C      'EXP'    135    10
C      'LN'     136    10
C      'LØG'    137    10
C      'INT'    138    10
C      'SIGN'   139    10
C
C      'THEN'   141
C      '('      142    1
C      ')'      143    2
C      APØST.   144
C      IF       145    11
C      TRUE     146    -1
C      SCØLØN   147    0
C      EQUAL    148    0
C
C      '.ØR.'   151    3
C      '.AND.'  152    4
C      '.NØT.'  153    5
C      '.GT.'   154    6
C      '.GE.'   155    6
C      '.LT.'   156    6
C      '.LE.'   157    6
C      '.EQ.'   158    6
C      '.NE.'   159    6
C
C      SKIPS ARE NUMBERED 161 - 169, LAST DIGIT INDICATES NØ. ØF SKIPS
C      D.E. EQUATIØN DEGREES ARE 171-179, LAST DIGIT INDICATES EQUATIØN
```

```
C     D. E. EQUATION NUMBER (ASSIGNED CONSEC. FROM 1) CODED 181-189
C
C
C      LTOG1 = 1          JUST RECOGNIZED HIGHEST DERIVATIVE OF X.
C                         NEXT INPUT SYMBOL NOT YET RECOGNIZED
C      LTOG2 = 1          JUST PROCESSED A /;/,          OR JUST STARTED
C      LTOG3 = 1          JUST PROCESSED A /;/, /THEN/, OR JUST STARTED
C      LTOG4 = 1          THIS IS NOT A BLANK LINE
C      LTOG5 = 1          HAVE RECOGNIZED THE DECIMAL POINT
C      LTOG6 = 1          DECODING A VARIABLE NAME
C      LTOG7 = 1          DECODING A NUMBER
C      LTOG8 = 1          DECODING A DIFFERENTIAL EQUATION
C      LTOG9 = 1          INTERPRETING A SKIP
C      LTOG10 = 1         EVALUATING A DIFFERENTIAL EQUATION
C      LTOG11 = 1         PROCESSING AN IF ARGUMENT
C
       SUBROUTINE COMP(ITEXT)
       REAL  Z(100)
       INTEGER STACKA(200),STKAH(200),STACKB(40 ),STKBH(40 )
       INTEGER STACKC(200), PSCT, PSCB, N
       INTEGER PSAB, PSAT, PSBT,PI1
       INTEGER INPUT(100)
       DOUBLE PRECISION ZT(10), ZDOT(10), DEL, TT
       DIMENSION ZI(9)
       INTEGER ITEXT(24,40)
       COMMON Z,ZT,ZDOT,DEL,TT,A,B,C,D,E,F,G,H,
      *XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,TI,TF,ZI,
      *XMIN,XMAX,YMIN,YMAX,XFACT,YFACT
       COMMON IDEV,NO,NN,LSTAT,N,STACKC,LFUNCT,L
       DATA LA,LB,LC,LD,LE,LF,LG,LH,LI,LJ,LK,LL,LM,LLN,
      *LO,LP,LQ,LR,LS,LT,LU,LV,LW,LX,LY,LZ
      */4HA   ,4HB   ,4HC   ,4HD   ,4HE   ,4HF   ,4HG   ,4HH   ,
      * 4HI   ,4HJ   ,4HK   ,4HL   ,4HM   ,4HN   ,4HO   ,4HP   ,
      * 4HQ   ,4HR   ,4HS   ,4HT   ,4HU   ,4HV   ,4HW   ,4HX   ,
      * 4HY   ,4HZ   /
```

```
      INTEGER PLUS,MINUS,DIV,MULT,LPAREN,RPAREN,APOS,BLANK,EQL
     *,SCOLON, PERIOD
      DATA PLUS,MINUS,DIV,MULT,LPAREN,RPAREN,APOS,BLANK,EQL
     *,SCOLON, PERIOD
     */4H+    ,4H-    ,4H/    ,4H*    ,4H(    ,4H)    ,4H'    ,4H     ,4H=    ,
     * 4H;    ,4H.    /
      INTEGER NO,N1,N2,N3,N4,N5,N6,N7,N8,N9
      DATA    NO,N1,N2,N3,N4,N5,N6,N7,N8,N9
     */4H0    ,4H1    ,4H2    ,4H3    ,4H4    ,4H5    ,4H6    ,4H7    ,4H8    ,
     *4H9    /
      OUTPUT(3)' INTO COMP '
      OUTPUT(3)A,B,C,D,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
     *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,N9,NN,LSTAT,N
      ENCODE(96,10,ITEXT(1,30))
   10 FORMAT(' COMPILING THE SYSTEM EQUATIONS')
      CALL TEXTO(IDEV,ITEXT(1,30),24,30,1,1,2,IER)
      WRITE(6,1200)
 1200 FORMAT('OTHE SYSTEM DESCRIPTION EQUATIONS AND/OR PROGRAM FOLLOWO')
      LWFP1 = 0
      LWFP1 = 1
      LWFP2 = 0
      LWFP2 = 1
      LWITP1 = 0
      LWITP1 = 1
      LWITP2 = 0
      LWITP2 = 1
      LWUSE = 0
      LWUSE = 1
      NAVAR = 0
      NDE = 0
      Z(10) = -1.0
      N = 10
      DO 692 I=1,200
      STKAH(I) = -1
  692 CONTINUE
```

```
      PSAT = 0
      LNUM = 4
C
C  READING IN DIFFERENTIAL EQUATIØN
C
  901 LNUM = LNUM + 1
      IF(PSAT.GT.0.AND.LWFP1.EQ.1) CALL DISPLY(STACKA,PSAT)
      DECØDE(96,1030,ITEXT(1,LNUM))(INPUT(I),I=1,96)
 1030 FØRMAT(96A1)
      LSTØP = 0
      LTØG1 = 0
      LTØG2 = 1
      LTØG3 = 1
      LTØG4 = 0
      LTØG5 = 0
      LTØG6 = 0
      LTØG7 = 0
      LTØG8 = 0
      LTØG9 = 0
      LTØG10 = 0
      LTØG11 = 0
      RNUM = 0.0
      NPAREN = 0
      WRITE(6,1120)(INPUT(I),I=1,96)
 1120 FØRMAT(' ',80A1)
C
C  BUILDING STACK A
C
      PI1 = 0
  605 PI1 = PI1 + 1
      IF(PI1.GT.96) GØ TØ 177
C
C  CHECKING FØR BLANK LINE INDICATING END ØF EQUATIØNS
C
      IF((LTØG4.EQ.0 .AND. PI1.LT.96) .ØR. ( LTØG4.EQ.1)) GØ TØ 607
```

```
      GO TO 201
C                                                              IF, HIGH X
  607 LTOG3 = 0
      IF(INPUT(PI1).EQ.BLANK) GO TO 605
      LTOG4 = 1
      IF(INPUT( PI1 ).NE. LI) GO TO 603
      IF(INPUT(PI1+1).NE. LF) GO TO 160
      LTOG11 = 1
      STACKA(PSAT+1) = 145
      STKAH (PSAT+1) = 11
      NPAREN = 0
      PI1 =PI1 + 1
      GO TO 152
  603 IF(INPUT(PI1).NE.LX) GO TO 101
      LTOG8 = 1
      NDE = NDE + 1
      N = 0
      DO 691 I=1,10
      IF(INPUT(PI1 + I).NE.APOS) GO TO 604
      N = N + 1
  691 CONTINUE
      WRITE(6,1050)
 1050 FORMAT(' RANK OF DIFFERENTIAL EQUATION IS TO HIGH, PLEASE CORRECT'
     *)
      GO TO 601
  606 WRITE(6,1060)
 1060 FORMAT(' THE HIGHEST DERIVATIVE OF X MUST NOT HAVE A COEFFICIENT',
     *', PLEASE CORRECT')
      GO TO 601
  604 IF(LTOG2.EQ.0) GO TO 608
      STACKA(PSAT+1) = 145
      STKAH(PSAT+1) = 10
      STACKA(PSAT+2) = 146
      STKAH (PSAT+2) = -1
      PSAT = PSAT + 2
```

```
  608 STACKA(PSAT+1) = 170 + N
      STKAH(PSAT+1) = -1
      PSAT = PSAT + 1
      IF(NDE.GT.9) GO TO 309
      STACKA(PSAT+1) = 180 + NDE
      STKAH(PSAT + 1) =  1
      PSAT = PSAT + 1
  309 STACKA(PSAT + 1) = 10
      STKAH(PSAT + 1) = -1
      STACKA(PSAT + 2) = 124
      STKAH(PSAT + 2) = 8
      STACKA(PSAT + 3) = 142
      STKAH(PSAT + 3) = 1
      PSAT = PSAT + 2
      PI1 = PI1 + I - 1
      NPAREN = 0
      LTOG1 = 1
      GO TO 134
C
 1011 PI1 = PI1 + 1
      IF(PI1.GE.96 ) GO TO 177
  101 IF(INPUT( PI1 ).EQ.BLANK) GO TO 1011
C                                                           DERIV. OF X
      IF(INPUT(PI1).NE.LX) GO TO 103
      NX = 0
      LAST = N
      IF(LTOG8.EQ.0) LAST = 9
      DO 197 I=1,LAST
      IF(INPUT(PI1 + I).NE.APOS) GO TO 102
      NX = NX + 1
  197 CONTINUE
      WRITE(6,1070)
 1070 FORMAT(' DERIVATIVE OF X TO HIGH, PLEASE CORRECT')
      GO TO 601
  102 STACKA(PSAT + 1) = I
```

```
      STKAH(PSAT + 1) = -1
      PI1 = PI1 + I - 1
      GO TO 152
C
C  RECOGNIZING THE BINARY OPERATORS
C                                                             PLUS
  103 IF(INPUT( PI1 ).NE. PLUS) GO TO 104
      STACKA(PSAT + 1) = 121
      STKAH(PSAT + 1) = 7
      GO TO 152
C                                                             MINUS
  104 IF(INPUT( PI1 ).NE.MINUS) GO TO 105
      STACKA(PSAT + 1) = 122
      STKAH(PSAT + 1) = 7
      GO TO 152
C                                                             DIV
  105 IF(INPUT( PI1 ).NE. DIV ) GO TO 106
      STACKA(PSAT + 1) = 123
      STKAH(PSAT + 1) = 8
      GO TO 134
C                                                             POWER
  106 IF(INPUT( PI1 ).NE. MULT) GO TO 107
      IF(INPUT(PI1+1).NE. MULT) GO TO 108
      STACKA(PSAT + 1) = 125
      STKAH(PSAT + 1) = 9
      PI1 = PI1 + 1
      GO TO 152
C                                                             MULT
  108 STACKA(PSAT + 1) = 124
      STKAH(PSAT + 1) = 8
      GO TO 134
C
C  RECOGNITION OF OPERANDS AND UNARY OPERATORS
C                                                             ABS
  107 IF(INPUT( PI1 ).NE.  LA ) GO TO 109
```

```
      IF(INPUT(PI1+1).NE.  LB ) GØ TØ 110
      IF(INPUT(PI1+2).NE.  LS ) GØ TØ 160
      STACKA(PSAT + 1) = 134
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GØ TØ 152
C                                                       A
  110 STACKA(PSAT + 1) = 11
      STKAH(PSAT + 1) = -1
      GØ TØ 152
C                                                       B
  109 IF(INPUT( PI1 ).NE.  LB ) GØ TØ 112
      STACKA(PSAT + 1) = 12
      STKAH(PSAT + 1) = -1
      GØ TØ 152
C                                                       CØS
  112 IF(INPUT( PI1 ).NE.  LC ) GØ TØ 113
      IF(INPUT(PI1+1).NE.  LØ ) GØ TØ 114
      IF(INPUT(PI1+2).NE.  LS ) GØ TØ 160
      STACKA(PSAT + 1) = 132
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GØ TØ 152
C                                                       C
  114 STACKA(PSAT + 1) = 13
      STKAH(PSAT + 1) = -1
      GØ TØ 152
C                                                       D
  113 IF(INPUT( PI1 ).NE.  LD ) GØ TØ 115
      STACKA(PSAT + 1) = 14
      STKAH(PSAT + 1) = -1
      GØ TØ 152
C                                                       EXP
  115 IF(INPUT( PI1 ).NE.  LE ) GØ TØ 116
      IF(INPUT(PI1+1).NE.  LX ) GØ TØ 117
```

```
      IF(INPUT(PI1+2).NE.  LP ) GØ TØ 160
      STACKA(PSAT + 1) = 135
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GØ TØ 152
C                                                        E
  117 STACKA(PSAT + 1) = 15
      STKAH(PSAT + 1) = -1
      GØ TØ 152
C                                                        F
  116 IF(INPUT( PI1 ).NE.  LF ) GØ TØ 118
      STACKA(PSAT + 1) = 16
      STKAH(PSAT + 1) = -1
      GØ TØ 152
C                                                        G
  118 IF(INPUT( PI1 ).NE.  LG ) GØ TØ 119
      STACKA(PSAT + 1) = 17
      STKAH(PSAT + 1) = -1
      GØ TØ 152
C                                                        H
  119 IF(INPUT( PI1 ).NE.  LH ) GØ TØ 3120
      STACKA(PSAT + 1) = 18
      STKAH(PSAT + 1) = -1
      GØ TØ 152
C                                                        INT
 3120 IF(INPUT( PI1 ).NE. LI) GØ TØ 120
      IF(INPUT(PI1+1).NE. LLN) GØ TØ 160
      IF(INPUT(PI1+2).NE. LT) GØ TØ 160
      STACKA(PSAT+1) = 138
      STKAH(PSAT+1) = 10
      PI1 = PI1 + 2
      GØ TØ 152
C                                                        TAN
  120 IF(INPUT( PI1 ).NE.  LT ) GØ TØ 121
      IF(INPUT(PI1+1).NE.  LA ) GØ TØ 122
```

```
      IF(INPUT(PI1+2).NE.  LLN) GØ TØ 160
      STACKA(PSAT + 1) = 133
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GØ TØ 152
C                                                           THEN
  122 IF(INPUT(PI1+1).NE.  LH ) GØ TØ 125
      IF(INPUT(PI1+2).NE.  LE ) GØ TØ 160
      IF(INPUT(PI1+3).NE.  LLN) GØ TØ 160
      PSAT = PSAT - 1
      IF(NPAREN.NE.0) GØ TØ 170
      PI1 = PI1 + 3
      LTØG2 = 0
      LTØG3 = 1
      LTØG11 = 0
      GØ TØ 152
C                                                           T
  125 STACKA(PSAT + 1) = 19
      STKAH(PSAT + 1) = -1
      GØ TØ 152
C                                                           SIN
  121 IF(INPUT( PI1 ).NE.  LS ) GØ TØ 123
      IF(INPUT(PI1+1).NE.  LI ) GØ TØ 701
      IF(INPUT(PI1+2).NE.  LLN) GØ TØ 3121
      STACKA(PSAT + 1) = 131
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GØ TØ 152
C                                                           SKIP
  701 IF(INPUT(PI1+1).NE.LK) GØ TØ 160
      IF(INPUT(PI1+2).NE.LI) GØ TØ 160
      IF(INPUT(PI1+3).NE.LP) GØ TØ 160
      PI1 = PI1 + 3
  702 PI1 = PI1 + 1
      IF(INPUT(PI1).EQ.BLANK) GØ TØ 702
```

```
      LTØG9 = 1
      GØ TØ 180
  703 LTØG9 = 0
      STACKA(PSAT+1) = 160 + RNUM + 1.5
      STKAH(PSAT+1) = -1
      PI1 = PI1 - 1
      GØ TØ 152
C                                                                 SIGN
 3121 IF(INPUT(PI1+2).NE. LG) GØ TØ 160
      IF(INPUT(PI1+3).NE.LLN) GØ TØ 160
      STACKA(PSAT+1) = 139
      STKAH(PSAT+1) = 10
      PI1 = PI1 + 3
      GØ TØ 152
C                                                                 LØG
  123 IF(INPUT( PI1 ).NE.  LL ) GØ TØ 126
      IF(INPUT(PI1+1).NE.  LØ ) GØ TØ 124
      IF(INPUT(PI1+2).NE.  LG ) GØ TØ 160
      STACKA(PSAT + 1) = 137
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 2
      GØ TØ 152
C                                                                 N
  124 IF(INPUT(PI1+1).NE.  LLN) GØ TØ 160
      STACKA(PSAT + 1) = 136
      STKAH(PSAT + 1) = 10
      PI1 = PI1 + 1
      GØ TØ 152
C
C
  170 WRITE(6,2220)
 2220 FØRMAT(' UNBALANCED PARENS IN LØGICAL EXPRESSIØN, PLEASE CØRRECT')
      GØ TØ 601
C                                                                 (
  126 IF(INPUT( PI1 ).NE.LPAREN) GØ TØ 127
```

```
      STACKA(PSAT + 1) = 142
      STKAH(PSAT + 1) = 1
      NPAREN = NPAREN + 1
C
C  TAKING CARE ØF THE UNARY ØPERATØRS √-√, AND √+√
C
  134 I = 1
  135 IF(INPUT(PI1+I).NE.BLANK) GØ TØ 136
      I = I + 1
      GØ TØ 135
  136 IF(INPUT(PI1+I).NE.MINUS) GØ TØ 137
      STACKA(PSAT+2) = 10
      STKAH(PSAT + 2) = -1
      STACKA(PSAT+3) = 124
      STKAH(PSAT + 3) = 8
      PSAT = PSAT + 2
      PI1 = PI1 + I
      LTØG1 = 0
      GØ TØ 152
  137 IF(INPUT(PI1+I).NE.PLUS ) GØ TØ 138
      PI1 = PI1 + I
      LTØG1 = 0
      GØ TØ 152
C
C  CHECK TØ SEE IF LEGAL THUS FAR
C
  138 IF(LTØG1.EQ.1) GØ TØ 606
      PI1 = PI1 + I - 1
      GØ TØ 152
C                                                          )
  127 IF(INPUT( PI1 ).NE.RPAREN) GØ TØ 128
      STACKA(PSAT + 1) = 143
      STKAH(PSAT + 1) = 2
      NPAREN = NPAREN - 1
      IF(NPAREN.NE.0) GØ TØ 152
```

```
      IF(LTOG11.NE.1) GO TO 152
      LTOG2 = 0
      LTOG3 = 1
      LTOG11 = 0
      GO TO 152
C
C  CHECK FOR EQUAL SIGN AND TAKE APROPRIATE ACTION IF FOUND
C                                                           =
  128 IF(INPUT(PI1).NE.EQL) GO TO 161
      IF(LTOG8.NE.0) GO TO 139
      STACKA(PSAT+1) = 148
      STKAH(PSAT+1) = 0
C
C  ACTION FOLLOWING J=J IN A NORMAL EQUATION
C
 2128 PI1 = PI1 + 1
      IF(INPUT(PI1).EQ.BLANK) GO TO 2128
      IF(INPUT(PI1).EQ.PLUS) GO TO 152
      PI1 = PI1 - 1
      IF(INPUT(PI1+1).NE.MINUS) GO TO 152
      PI1 = PI1 + 1
      STACKA(PSAT+2) = 10
      STKAH(PSAT+2) = -1
      STACKA(PSAT+3) = 124
      STKAH(PSAT+3) = 8
      PSAT = PSAT + 2
      GO TO 152
C
C  ACTION FOLLOWING J=J IN A DIFFERENTIAL EQUATION
C
  139 STACKA(PSAT + 1) = 143
      STKAH(PSAT + 1) = 2
      I = 1
  129 IF(INPUT(PI1+I).NE.BLANK) GO TO 130
      I = I + 1
```

```
      GO TO 129
  130 IF(INPUT(PI1+I).NE.MINUS) GO TO 131
      STACKA(PSAT+2) = 122
      STKAH(PSAT + 2) = 7
      PI1 = PI1 + 1
      GO TO 133
  131 IF(INPUT(PI1+I).NE.PLUS ) GO TO 132
      PI1 = PI1 + 1
  132 STACKA(PSAT+2) = 121
      STKAH(PSAT + 2) = 7
  133 PI1 = PI1 + I - 1
      PSAT = PSAT + 1
      GO TO 152
C
C                                                            .
  161 IF(INPUT( PI1 ).NE.PERIOD) GO TO 175
      IF(LTOG3.EQ.1) GO TO 155
C                                                            .OR.
      IF(INPUT(PI1+1).NE.LO) GO TO 162
      IF(INPUT(PI1+2).NE.LR) GO TO 160
      IF(INPUT(PI1+3).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 151
      STKAH(PSAT + 1) = 3
      PI1 = PI1 + 3
      GO TO 134
C                                                            .AND.
  162 IF(INPUT(PI1+1).NE.LA) GO TO 163
      IF(INPUT(PI1+2).NE.LLN) GO TO 160
      IF(INPUT(PI1+3).NE.LD) GO TO 160
      IF(INPUT(PI1+4).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 152
      STKAH(PSAT + 1) = 4
      PI1 = PI1 + 4
      GO TO 134
C                                                            .NOT.
```

```
  163 IF(INPUT(PI1+1).NE.LLN) GO TO 165
      IF(INPUT(PI1+2).NE.LO) GO TO 164
      IF(INPUT(PI1+3).NE.LT) GO TO 160
      IF(INPUT(PI1+4).NE.PERIOD) GO TO 160
      IF(STACKA(PSAT).NE.52) GO TO 171
      STACKA(PSAT + 1) = 153
      STKAH(PSAT + 1) = 5
      PI1 = PI1 + 4
      GO TO 134
C
  171 WRITE(6,2250)
 2250 FORMAT(' .NOT. CAN ONLY BE USED FOLLOWING .AND., PLEASE CORRECT')
      GO TO 601
C                                                                .LE.
  164 IF(INPUT(PI1+2).NE.LE) GO TO 160
      IF(INPUT(PI1+3).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 159
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GO TO 134
C                                                                .GT.
  165 IF(INPUT(PI1+1).NE.LG) GO TO 167
      IF(INPUT(PI1+2).NE.LT) GO TO 166
      IF(INPUT(PI1+3).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 154
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GO TO 134
C                                                                .GE.
  166 IF(INPUT(PI1+2).NE.LE) GO TO 160
      IF(INPUT(PI1+3).NE.PERIOD) GO TO 160
      STACKA(PSAT + 1) = 155
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GO TO 134
```

```
C                                                                     .LT.
  167 IF(INPUT(PI1+1).NE.LL) GØ TØ 169
      IF(INPUT(PI1+2).NE.LT) GØ TØ 168
      IF(INPUT(PI1+3).NE.PERIØD) GØ TØ 160
      STACKA(PSAT + 1) = 156
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GØ TØ 134
C                                                                     .LE.
  168 IF(INPUT(PI1+2).NE.LE) GØ TØ 160
      IF(INPUT(PI1+3).NE.PERIØD) GØ TØ 160
      STACKA(PSAT + 1) = 157
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GØ TØ 134
C                                                                     .EQ.
  169 IF(INPUT(PI1+1).NE.LE) GØ TØ 160
      IF(INPUT(PI1+2).NE.LQ) GØ TØ 160
      IF(INPUT(PI1+3).NE.PERIØD) GØ TØ 160
      STACKA(PSAT + 1) = 158
      STKAH(PSAT + 1) = 6
      PI1 = PI1 + 3
      GØ TØ 134
C
C  DECØDING VARIABLES USED BY USER
C                                                                    V
  175 IF(INPUT(PI1).NE.LV) GØ TØ 180
      PI1 = PI1 + 1
      LTØG6 = 1
      GØ TØ 180
  501 IF(RNUM.GT.0.5 .AND. RNUM.LT.40.5) GØ TØ 502
      WRITE(6,5501)
 5501 FØRMAT('0THE VARIABLE NUMBER IS ØUT ØF THE ALLØWED RANGE,'
     *' PLEASE CØRRECT')
      GØ TØ 601
```

```
  502 IRNUM = RNUM + 0.5
      STACKA(PSAT+1) = 60 + IRNUM
      STKAH(PSAT+1) = -1
      Z(60 + IRNUM) = IRNUM
      PI1 = PI1 - 1
      GØ TØ 152
C
C  RECØGNIZING NUMBERS
C
  180 IF(PI1.GT.96) GØ TØ 160
      IF(INPUT(PI1) .NE. NO) GØ TØ 181
      INTEG = 0
      GØ TØ 191
  181 IF(INPUT(PI1) .NE. N1) GØ TØ 182
      INTEG = 1
      GØ TØ 191
  182 IF(INPUT(PI1) .NE. N2) GØ TØ 183
      INTEG = 2
      GØ TØ 191
  183 IF(INPUT(PI1) .NE. N3) GØ TØ 184
      INTEG = 3
      GØ TØ 191
  184 IF(INPUT(PI1) .NE. N4) GØ TØ 185
      INTEG = 4
      GØ TØ 191
  185 IF(INPUT(PI1) .NE. N5) GØ TØ 186
      INTEG = 5
      GØ TØ 191
  186 IF(INPUT(PI1) .NE. N6) GØ TØ 187
      INTEG = 6
      GØ TØ 191
  187 IF(INPUT(PI1) .NE. N7) GØ TØ 188
      INTEG = 7
      GØ TØ 191
  188 IF(INPUT(PI1) .NE. N8) GØ TØ 189
```

NUMBERS

```
      INTEG = 8
      GØ TØ 191
  189 IF(INPUT(PI1) .NE. N9) GØ TØ 190
      ISTEG = 9
      GØ TØ 191
  190 IF(INPUT(PI1) .NE. PERIØD) GØ TØ 193
      IF(LTØG6.EQ.1) GØ TØ 501
      NDEC = 1
      LTØG5 = 1
      PI1 = PI1 + 1
      GØ TØ 180
C                                                       L ØF D. PNT
  191 IF(LTØG5.EQ.1) GØ TØ 192
      RNUM = RNUM*10.0 + INTEG
      PI1 = PI1 + 1
      LTØG7 = 1
      GØ TØ 180
C                                                       R ØF D. PNT
  192 RNUM = RNUM + INTEG/(10.0**NDEC)
      NDEC = NDEC + 1
      PI1 = PI1 + 1
      GØ TØ 180
  193 CØNTINUE
      IF(LTØG9.EQ.1) GØ TØ 703
      IF(LTØG6.EQ.1) GØ TØ 501
      IF(LTØG7.NE.1) GØ TØ 176
      IF(LTØG5.NE.0) GØ TØ 194
      IF(STACKA( PSAT ) .NE.125) GØ TØ 195
      STACKA( PSAT ) = 126
      GØ TØ 194
 195  IF(STACKA(PSAT-1).NE.125) GØ TØ 194
      IF (STACKA( PSAT ).NE.122) GØ TØ 196
      STACKA(PSAT-1) = 127
      PSAT = PSAT - 1
      GØ TØ 194
```

```
  196 IF(STACKA( PSAT ).NE.121) GO TO 160
      STACKA(PSAT-1) = 126
      PSAT = PSAT - 1
  194 STACKA(PSAT+1) = 20 + NAVAR + 1
      STKAH(PSAT+1) = -1
      Z(20 + NAVAR + 1) = RNUM
      NAVAR = NAVAR +1
      PI1 = PI1 - 1
      GO TO 152
C
  176 IF(INPUT(PI1).NE.SCOLON) GO TO 160
  177 LTOG8 = 0
      LTOG1 = 0
      LTOG2 = 1
      LTOG3 = 1
      LTOG4 = 0
      LTOG5 = 0
      LTOG6 = 0
      LTOG7 = 0
      LTOG9 = 0
      LTOG10 = 0
      STACKA(PSAT + 1) = 147
      STKAH(PSAT+1) = -1
      PSAT = PSAT + 1
C
  156 IF(NPAREN.EQ.0) GO TO 157
      WRITE(6,2230)
 2230 FORMAT(' UNBALANCED PARENS, PLEASE CORRECT')
      GO TO 601
  157 PI1 = PI1 + 1
      IF(PI1.LT.96 ) GO TO 761
      WRITE(3,1141)
 1141 FORMAT(100X,' FIRST PASS OK')
      GO TO 901
  761 IF(INPUT(PI1).EQ.BLANK) GO TO 157
```

```
      PSAT = PSAT - 1
      PI1 = PI1 - 1
      GO TO 152
C
  160 WRITE(6,1140)PI1
 1140 FORMAT(' WRONG SPELLING,CODE, OR J;J MISSING, CURRENT INPUT CHARAC
     *TER IS ',I5,'  PLEASE CORRECT ')
      GO TO 601
C
C  PREPARING TO RECOGNIZE NEXT INPUT CHARACTER
C
  152 PSAT = PSAT + 1
  151 PI1 = PI1 + 1
      IF(LWFP2.EQ.1) CALL DISPLY(STACKA,PSAT)
      RNUM = 0.0
      LTOG5 = 0
      LTOG6 = 0
      LTOG7 = 0
      IF(PI1.GT.96 ) GO TO 160
      IF(LTOG3.EQ.1) GO TO 607
      GO TO 101
C
  155 WRITE(3,2210)
 2210 FORMAT(' J;J, OR JWHENJ MISSING, OR USE OF LOGICAL OPERATORS ',/,
     *'        IN THE DIFFERENTIAL EQUATION, PLEASE CORRECT')
      GO TO 601
C
C  CONVERSION OF INFIX TO POLISH
C
  201 PSAB = 1
      PSCT = 0
      PSBT = 0
      WRITE(3,1801)
 1801 FORMAT('OFOLLOWING IS THE CODED EQUIVALENT OF THE INPUT STRINGO')
      WRITE(3,1800)(STACKA(I),I=1,PSAT)
```

```
      WRITE(3,1802)
 1802 FORMAT('OFOLLOWING IS THE CORRESPONDING HEIARCHIES ASSIGNEDO')
      WRITE(3,1800)(STKAH(I),I=1,PSAT)
 1800 FORMAT(20I6)
      WRITE(3,1803)
 1803 FORMAT('OFOLLOWING IS THE MNEMONIC DECODE OF THE NUMERICALLY',
     *' CODED INPUT STRINGO')
      CALL DISPLY(STACKA,PSAT)
      WRITE(3,1870)
 1870 FORMAT('0')
      WRITE(3,1820)
 1820 FORMAT(' STARTING CONVERSION TO POLISH')
C
  202 CONTINUE
      IF(LWITP1.EQ.0)GO TO 3201
 3202 IF(PSBT.GT.0) CALL DISPLY(STACKB,PSBT)
      IF(PSCT.GT.0) CALL DISPLY(STACKC,PSCT)
 3201 IF(PSAB.GT.PSAT)  GO TO 300
      IF(STKAH(PSAB).GE.0) GO TO 203
      STACKC(PSCT+1) = STACKA(PSAB)
      PSCT = PSCT + 1
      PSAB = PSAB + 1
      IF(LWITP2.EQ.0) GO TO 204
 2111 WRITE(3,2110)
      CALL DISPLY(STACKC,PSCT)
C
 2110 FORMAT(' FOLLOWING 202')
      CALL DISPLY(STACKB,PSBT)
  204 IF(PSBT.EQ.0) GO TO 202
      IF(STKBH(PSBT).LT.STKAH(PSAB)) GO TO 202
      STACKC(PSCT+1) = STACKB(PSBT)
      PSCT = PSCT + 1
      PSBT = PSBT - 1
      IF(LWITP2.EQ.0) GO TO 204
 2121 WRITE(3,2120)
```

```
 2120 FORMAT(' FOLLOWING 204')
      CALL DISPLY(STACKB,PSBT)
      CALL DISPLY(STACKC,PSCT)
      GO TO 204
C
  203 IF(STACKA(PSAB).EQ.143) GO TO 205
      STACKB(PSBT+1) = STACKA(PSAB)
      STKBH(PSBT+1) = STKAH(PSAB)
      PSAB = PSAB + 1
      PSBT = PSBT + 1
      IF(LWITP2.EQ.0) GO TO 202
 2131 WRITE(3,2130)
 2130 FORMAT(' FOLLOWING 203')
      CALL DISPLY(STACKB,PSBT)
      CALL DISPLY(STACKC,PSCT)
      GO TO 202
C
  205 IF(STACKB(PSBT).NE.142) GO TO 211
      PSAB = PSAB + 1
      PSBT = PSBT - 1
      IF(LWITP2.EQ.0) GO TO 204
 2141 WRITE(3,2140)
 2140 FORMAT(' FOLLOWING 205')
      CALL DISPLY(STACKB,PSBT)
      CALL DISPLY(STACKC,PSCT)
      GO TO 204
  211 WRITE(6,1080)
 1080 FORMAT(' DID NOT FIND /(/ WHEN EXPECTED, PLEASE CHECK INPUT STRING
     * BETWEEN YOUR PARENS ')
      GO TO 601
C
C
C  PRINTING RESULTS OF COMPILATION PHASE
C
  300 WRITE(3,1081)
```

```
1081 FORMAT('OFINISHED COMPILATION, THE DECODED POLISH STACK ISO')
     CALL DISPLY(STACKC,PSCT)
     WRITE(3,1831)
1831 FORMAT('O THE COMPUTER ASSIGNED VARIABLES AREO')
     DO 393 I = 1,NAVAR
     WRITE(3,2231)I, Z(20 + I)
2231 FORMAT(' C(',I2,') = ',F15.5)
 393 CONTINUE
     LSTAT = 0
     NN = PSCT
     OUTPUT(3)' LEAVING COMP '
     OUTPUT(3)A,B,C,D,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
    *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,NO,NN,LSTAT,N
     OUTPUT(3) DEL,T
     WRITE(3,8110)Z,ZT,ZDOT
     WRITE(3,8111)STACKC
     ENCODE(96,11,ITEXT(1,30))
  11 FORMAT('                                                          ')
     CALL TEXTO(IDEV,ITEXT(1,30),24,30,1,1,2,IER)
     RETURN
 601 CONTINUE
     LSTAT = 2
     OUTPUT(3)' LEAVING COMP WITH ERROR '
     OUTPUT(3)A,B,C,D,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
    *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,NO,NN,LSTAT,N
     OUTPUT(3) DEL,T
     WRITE(3,8110)Z,ZT,ZDOT
     WRITE(3,8111)STACKC
8110 FORMAT(5X,10F12.3)
8111 FORMAT(5X,10I12)
     ENCODE(96,11,ITEXT(1,30))
     CALL TEXTO(IDEV,ITEXT(1,30),24,30,1,1,2,IER)
     RETURN
     END
```

```
      SUBROUTINE DISPLY (LISTI,N)
      DIMENSION LISTI(1  ), LIST8(200)
      INTEGER ANUM1(20),ANUM2(40),ANUM3(40),ANUM4(40),ANUM5(30)
      DATA ANUM1/
     *4HZ(1),4HZ(2),4HZ(3),4HZ(4),4HZ(5),4HZ(6),4HZ(7),4HZ(8),4HZ(9),
     *4H-1.0,4HA   ,4HB   ,4HC   ,4HD   ,4HE   ,4HF   ,4HG   ,4HH   ,
     *4HT   ,4HO   /
      DATA ANUM2/
     *            4HC1  ,4HC2  ,4HC3  ,4HC4  ,4HC5  ,4HC6  ,4HC7  ,
     *4HC8  ,4HC9  ,4HC10 ,4HC11 ,4HC12 ,4HC13 ,4HC14 ,4HC15 ,4HC16 ,
     *4HC17 ,4HC18 ,4HC19 ,4HC20 ,4HC21 ,4HC22 ,4HC23 ,4HC24 ,4HC25 ,
     *4HC26 ,4HC27 ,4HC28 ,4HC29 ,4HC30 ,4HC31 ,4HC32 ,4HC33 ,4HC34 ,
     *4HC35 ,4HC36 ,4HC37 ,4HC38 ,4HC39 ,4HC40 /
      DATA ANUM3/
     *4HV1  ,4HV2  ,4HV3  ,4HV4  ,4HV5  ,4HV6  ,4HV7  ,4HV8  ,4HV9  ,
     *4HV10 ,4HV11 ,4HV12 ,4HV13 ,4HV14 ,4HV15 ,4HV16 ,4HV17 ,4HV18 ,
     *4HV19 ,4HV20 ,4HV21 ,4HV22 ,4HV23 ,4HV24 ,4HV25 ,4HV26 ,4HV27 ,
     *4HV28 ,4HV29 ,4HV30 ,4HV31 ,4HV32 ,4HV33 ,4HV34 ,4HV35 ,4HV36 ,
     *4HV37 ,4HV38 ,4HV39 ,4HV40 /
      DATA ANUM4/
     *4H+   ,4H-   ,4H/   ,4H*   ,4H**R ,4H**PI,4H**NI,4HO   ,4HO   ,
     *4HO   ,4HSIN ,4HCOS ,4HTAN ,4HABS ,4HEXP ,4HLN  ,4HLOG ,4HINT ,
     *4HSIGN,4HO   ,4HTHEN,4H(   ,4H)   ,4H'   ,4HIF  ,4HTRUE,4H;   ,
     *4H=   ,4HO   ,4HO   ,4HOR  ,4HAND ,4HNOT ,4HGT  ,4HGE  ,4HLT  ,
     *4HLE  ,4HEQ  ,4HNE  ,4HO   /
      DATA ANUM5/
     *                        4HSK1 ,4HSK2 ,4HSK3 ,4HSK4 ,4HSK5 ,
     *4HSK6 ,4HSK7 ,4HSK8 ,4HSK9 ,4HO   ,4HN=1 ,4HN=2 ,4HN=3 ,4HN=4 ,
     *4HN=5 ,4HN=6 ,4HN=7 ,4HN=8 ,4HN=9 ,4H    ,4HL=1 ,4HL=2 ,
     *4HL=3 ,4HL=4 ,4HL=5 ,4HL=6 ,4HL=7 ,4HL=8 ,4HL=9 ,4HTIME/
      IF(N.EQ.0) GO TO 99
      DO 91 I=1,N
      NUMB=LISTI(I)
      IF(NUMB.GT.20) GO TO 11
      LIST8(I) = ANUM1(NUMB)
```

```
      GØ TØ 91
   11 NUMB = NUMB - 20
      IF(NUMB.GT.40) GØ TØ 12
      LISTØ(I) = ANUM2(NUMB)
      GØ TØ 91
   12 NUMB = NUMB - 40
      IF(NUMB.GT.40) GØ TØ 14
      LISTØ(I) = ANUM3(NUMB)
      GØ TØ 91
   14 NUMB = NUMB - 60
      IF(NUMB.GT.40) GØ TØ 15
      LISTØ(I) = ANUM4(NUMB)
      GØ TØ 91
   15 NUMB = NUMB - 40
      IF(NUMB.GT.30) GØ TØ 13
      LISTØ(I) = ANUM5(NUMB)
      GØ TØ 91
   13 WRITE(3,140)LISTI(I)
  140 FØRMAT('0 CØDE NØ. IS ØUT ØF ALLØWED RANGE, IT IS0',I5)
      RETURN
   91 CØNTINUE
      WRITE(3,100)(LISTØ(I),I=1,N)
  100 FØRMAT('0', 20A6)
      WRITE(3,110)
  110 FØRMAT('0')
      RETURN
   99 WRITE(3,120)N
  120 FØRMAT(' N=',I5)
      RETURN
      END
```

```
C
C  THIS SUBROUTINE PLOTS THE X-Y GRID AND DETERMINES THE
C  VARIOUS SCALE FACTORS THAT ARE USED IN THIS SUBROUTINE
C  AND IN FOLLOWING SUBROUTINES ESPECIALLY FOR
C  NORMALIZATION.
      SUBROUTINE GRID(ITDIR,IGDIR,ITEXT,IMAGE)
      DOUBLE PRECISION Z(10), ZDOT(10), DEL, T
      DIMENSION ZI(9)
      INTEGER ITDIR(41),IGDIR(5),ITEXT(24,40),IMAGE(4812)
      INTEGER STACKC(200)
      DATA LABEL/4H    /
      DOUBLE PRECISION ITITLE(12)
      DIMENSION ZC(100)
      COMMON ZC,Z,ZDOT,DEL,T,A,B,C,D,E,F,G,H,
     *XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,TI,TF,ZI,
     *XMIN,XMAX,YMIN,YMAX,XFACT,YFACT
      COMMON IDEV,NO,NN,LSTAT,N,STACKC,LFUNCT,L
      OUTPUT(3)' INTO GRID '
      OUTPUT(3)A,B,C,D,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
     *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,NO,NN,LSTAT,N
      OUTPUT(3) DEL,T
      WRITE(3,8110)ZC,Z,ZDOT
      WRITE(3,8111)STACKC
 8110 FORMAT(5X,10F12.3)
 8111 FORMAT(5X,10I12)
      ENCODE(96,10,ITEXT(1,30))
   10 FORMAT(' COMPUTING  SLOPES')
      CALL TEXTO(IDEV,ITEXT(1,30),24,30,1,1,2,IER)
      XMIN=XCENT-XSCALE*NXSIZE/2.
      XMAX=XCENT+XSCALE*NXSIZE/2.
      YMIN=YCENT-YSCALE*NYSIZE/2.
      YMAX=YCENT+YSCALE*NYSIZE/2.
      IYRIGH=(XCENT-XMIN)/XSCALE + 0.5
      IMAGE(1)=IHEAD(0,7)
      X=XMIN
```

```
      IMAGE(2)=IPACK(X,Y,0)
      X=XMAX
      Y=0.0
      CALL NØRM(1,X,Y,X,Y,1)
      IMAGE(3)=IPACK(X,Y,1)
      X=0.0
      Y=YMAX
      CALL NØRM(1,X,Y,X,Y,1)
      IMAGE(4)=IPACK(X,Y,0)
      X=0.0
      Y=YMIN
      CALL NØRM(1,X,Y,X,Y,1)
      IMAGE(5)=IPACK(X,Y,1)
      IMAGE(6)=IPACK(X,Y,0)
   71 CALL GRAPHØ(IDEV,IMAGE,6,2,IER)
      IF(IER.NE.0)CALL ERRØR(4HGRID,4H 71 ,0,0)
      ØUTPUT(3)' LEAVING GRID '
      ØUTPUT(3)A,B,C,D,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
     *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,NØ,NN,LSTAT,N
      ØUTPUT(3) DEL,T
      WRITE(3,8110)ZC,Z,ZDØT
      WRITE(3,8111)STACKC
      RETURN
      END
```

```
      MM=1
      DØ 91  IX=1,NX
      Z(1)=XMIN+IX*XSCALE/NØ
      DØ 91  IY=1,NY
      I=I+2
      Z(2)=YMIN+IY*YSCALE/NØ
      CALL  INTER
      IF(N.GT.2) II  = 2
      IF(N.GT.2) GØ TØ 93
      IF(LFUNCT.EQ.1) I=I-2; GØ TØ 91
      IF(DABS(Z(2)).GT.1.E-20) GØ TØ 25
      THETA = 3.14159/2.
      GØ TØ 32
   25 SLØPE = (ZDØT(2)/YSCALE)/(Z(2)/XSCALE)
      THETA=ATAN(SLØPE)
   32 AMAGX = SIZE*XSCALE*CØS(THETA)
      AMAGY = SIZE*YSCALE*SIN(THETA)
      X(1)=Z(1)+AMAGX
      X(2)=Z(1)-AMAGX
      Y(1)=Z(2)+AMAGY
      Y(2)=Z(2)-AMAGY
      CALL NØRM(2,X,Y,X,Y,1)
      IMAGE(I)=IPACK(X(1),Y(1),0)
      IMAGE(I+1)=IPACK(X(2),Y(2),MM)
 91   CØNTINUE
      I=I+1
      II=I+1
   93 CØNTINUE
      DØ 92 K = II,1060
      IMAGE(K)=IPACK(X(1),Y(1),0)
   92 CØNTINUE
   71 CALL GRAPHØ(IDEV,IMAGE,1060,3,IER)
      IF(IER.NE.0)CALL ERRØR(4HSLØP,4H 7U ,I,J)
      ØUTPUT(3)' LEAVING SLØPES '
      ØUTPUT(3)A,B,C,D,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
```

```
     *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,NO,NN,LSTAT,N
      OUTPUT(3) DEL,T
      WRITE(3,8110)ZC,Z,ZDOT
      WRITE(3,8111)STACKC
      ENCODE(96,11,ITEXT(1,30))
   11 FORMAT('                                                          ')
      CALL TEXTO(IDEV,ITEXT(1,30),24,30,1,1,2,IER)
   99 RETURN
      END
```

```
C
C
C
C  THIS SUBROUTINE SOLVES THE ACTUAL SYSTEM AND SENDS THE SOLUTION
C  OUT TO THE ADAGE AND ALSO GIVES HARD PHASE PLANE AND TIME RESPONSE
C  PLOTS ON THE PRINTER.  IN ADDITION ALL OF THE SYSTEM AND SOLUTION
C  PARAMETERS ARE PRINTED SO ONE KNOWS WHAT WENT INTO THE PLOTS.
C
C
      SUBROUTINE SOLVE(ITDIR,IGDIR,ITEXT,D,IMSOLV)
      DIMENSION ZI(9), D(401,6), JXY(6)
      DATA LABEL/4H    /
      INTEGER IMSOLV(402)
      DOUBLE PRECISION Z(10), ZDOT(10), DEL, T
      INTEGER ITDIR(41),IGDIR(5),ITEXT(24,40)
      INTEGER STACKC(200)
      DIMENSION ZC(100)
      COMMON ZC,Z,ZDOT,DEL,T,A,B,C,DD,E,F,G,H,
     *XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,TI,TF,ZI,
     *XMIN,XMAX,YMIN,YMAX,XFACT,YFACT
      COMMON IDEV,NO,NN,LSTAT,N,STACKC,LFUNCT,L
      OUTPUT(3)' INTO SOLVE '
      OUTPUT(3)A,B,C,DD,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
     *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,NO,NN,LSTAT,N
      OUTPUT(3) DEL,T
      WRITE(3,8110)ZC,Z,ZDOT
      WRITE(3,8111)STACKC
 8110 FORMAT(5X,10F12.3)
 8111 FORMAT(5X,10I12)
      WRITE(6,910)(((ITEXT(I,J)),I=1,24),J=1,40)
  910 FORMAT('1 THE GRAPHICS SCREEN DATA IS:',/,(24A4))
      WRITE(1,30)
   30 FORMAT(//' AREA     TIME              Z(1)              Z(2)          '//)
      IMSOLV(1) = IHEAD(0,7)
      DO 93 I = 1,401
```

```
      IMSOLV(I) = IPACK(0.0,0.0,0)
   93 CONTINUE
      NT=0
      LR = 0
      MM= ((TF-TI)/DEL)/40
      T=TI
      D(1,1) = TI
      DO 91 J=1,5
      Z(J) = ZI(J)
      D(1,J+1) = ZI(J)
   91 CONTINUE
      DO 92 I=3,8
   92 Z(I)=0.0
      D(1,1)=ZI(1)
      D(1,2)=ZI(2)
      D(1,3)=T
      CALL NORM(1,ZI(1),ZI(2),XN,YN,1)
      IMSOLV(2)=IPACK(XN,YN,0)
      K = 1
      I=1
      X1 = ZI(1)
      Y1 = ZI(2)
      AAA=1.0E08
   18 I=I+1
   12 CALL INTER
      IF(LSTAT.EQ.3) RETURN
      DO 194 ILG=1,N
      IF(ZDOT(ILG).GT.AAA)ZDOT(ILG)=AAA
      IF(ZDOT(ILG).LE.-AAA)ZDOT(ILG)=-AAA
  194 CONTINUE
      S=RKLDEQ(N,Z,ZDOT,T,DEL,NT)
   50 FORMAT(6F15.5)
      IF(S-1.0)11,12,14
   11 STOP
   14 CONTINUE
```

```
C
C  CHECKING FOR WILD SOLUTION
C
      DO 191 ILG=1,N
      IF(Z(ILG).GT.1.0E10)Z(ILG)=1.0E10
      IF(Z(ILG).LT.-1.0E10)Z(ILG)=-1.0E10
  191 CONTINUE
      X=Z(1)
      Y=Z(2)
      D(I,1)=X
      D(I,2)=Y
      D(I,3)=T
      Z3=Z(3)
      Z4=Z(4)
      Z5=Z(5)
      D(I,4)=Z3
      D(I,5)=Z4
      D(I,6)=Z5
      CALL INTER
      IF(LSTAT.EQ.3) RETURN
      DELT = ((X-X1)*XFACT)**2 + ((Y-Y1)*YFACT)**2
      K = I+1
      CALL NORM(     1,D(I,1),D(I,2),U      ,V      ,1)
      IMSOLV(I+1)=IPACK(U,V,1)
      NSKIP=NSKIP+1
      IF(NSKIP.EQ.9) GO TO 709
      IF(DELT.LT.DIFF)GO TO 21
  709 CALL GRAPH9(IDEV,IMSOLV,400,4,IER)
      NSKIP=0
      X1 = X
      Y1 = Y
   21 IF(MOD(I,10).EQ.0)WRITE(1,20)L,T,(Z(M),M=1,N)
      IF(F.NE.0.0)       WRITE(4,20)L,T,(Z(M),M=1,N)
   20 FORMAT(I4,9F15.5)
      IF(400-I)15,15,16
```

```
15 WRITE(6,10)T
   ENCØDE(96,10,ITEXT(1,34))T
   CALL TEXTØ(IDEV,ITEXT(1,34),24,34,1,1,2,IER)
10 FØRMAT(' SØLUTIØN TERMINATED AT TIME =',F10.5,
  *'   WITH 400 SØLUTIØN PØINTS')
   GØ TØ 17
16 IF((TF - T)*DEL) 17,17,18
17 CØNTINUE
   CALL GRAPHØ(IDEV,IMSØLV,400,4,IER)
   JXY(1)=1
   JXY(2)=2
   CALL VPLØT(D,JXY,I,401,1,0,0.,0.,0.,0.)
   JXY(1)=3
   JXY(2)=1
   CALL VPLØT(D,JXY,I,401,1,0,0.,0.,0.,0.)
   DØ 192 J=4,N + 1
   JXY(2)=J
192 CALL VPLØT(D,JXY,I,401,1,0,0.,0.,0.,0.)
   ØUTPUT(3)' LEAVING SØLVE '
   ØUTPUT(3)A,B,C,DD,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
  *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,NØ,NN,LSTAT,N
   ØUTPUT(3) DEL,T
   WRITE(3,8110)ZC,Z,ZDØT
   WRITE(3,8111)STACKC
   RETURN
   END
```

```
C
C  THE FØLLØWING SUBRØUTINE IS PART ØF THE IBM/360 LIBRARY.
C  IT SØLVES THE DIFFERENTIAL EQUATIØN USING THE RUNGE-KUTTA METHØD.
C  THE DERIVATIVES NEEDED ARE SUPPLIED BY THE SYSTEM EQUATIØNS
C  VIA THE INTERPRETER.
C
      FUNCTIØN RKLDEQ (N,Y,F,X,H,NT)
      DØUBLE PRECISIØN Y,F,X,H,Q,H1,H2,H3,H6
      DIMENSIØN Y(2), F(2), Q(25)
C
      NT = NT +1
      GØ TØ (1,2,3,4),NT
    1 H1 = H
      H2 = H1 * .5DO
      H3 = H1 * 2.DO
      H6 = H1/6.DO
      DØ 11 J =1,N
   11 Q(J) = 0.DO
      A = .5DO
      X = X + H2
      GØ TØ 5
C
    2 A = .2928932188134525
      GØ TØ 5
C
    3 A = 1.7071067811865475
      X = X + H2
      GØ TØ 5
C
    4 DØ 41 I = 1,N
   41 Y(I) = Y(I) + H6 * F(I) -Q(I)/3.DO
      NT = 0
      RKLDEQ =2.
      GØ TØ 6
C
```

```
    5 DO 51 L = 1,N
      Y(L) = Y(L) + A *(H * F(L) -Q(L))
   51 Q(L) = H3 * A *F(L) +(1.DO-3.DO*A) *Q(L)
      RKLDEQ =1.
C
    6 RETURN
      END
```

```
      SUBR0UTINE INTER
      L0GICAL STACKL(30)
      REAL STACKW(50), STACKA(50)
      INTEGER PSWT,PSLT,PSCT,PSCB,STACKC(200)
      D0UBLE PRECISI0N ZT(10), ZD0T(10), DEL, TT
      DIMENSI0N ZI(9), Z(100)
      C0MM0N Z,ZT,ZD0T,DEL,TT,A,B,C,D,E,F,G,H,
     *XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,TI,TF,ZI,
     *XMIN,XMAX,YMIN,YMAX,XFACT,YFACT
      C0MM0N IDEV,N0,NN,LSTAT,N,STACKC,LFUNCT,L
      DATA M,LST0P/0,0/
C
      IF(M.GE.2) G0 T0 3192
      0UTPUT(3)' INT0 INTER '
      0UTPUT(3)A,B,C,D,E,F,G,H,XSCALE,YSCALE,XCENT,YCENT,NXSIZE,NYSIZE,
     *TI,TF,ZI,XMIN,XMAX,YMIN,YMAX,XFACT,YFACT,N0,NN,LSTAT,N
      WRITE(3,8110)Z,ZT,ZD0T
      WRITE(3,8111)STACKC
 8110 F0RMAT(5X,10F12.3)
 8111 F0RMAT(5X,10I12)
 3192 C0NTINUE
C
C
      PSCT = NN
      D0 3191 I = 1,9
      Z(I) = ZT(I)
 3191 C0NTINUE
      M=M+1
      Z(10) = -1.0
      T = TT
      Z(11) = A
      Z(12) = B
      Z(13) = C
      Z(14) = D
      Z(15) = E
```

```
      Z(16) = F
      Z(17) = G
      Z(18) = H
      Z(19) = T
      LFUNCT = 0
      LT0G10 = 0
      LWUSE = 0
      LWUSE = 1
C
C  USING THE P0LISH
C
  306 PSCB = 0
      PSWT = 0
      PSLT = 0
      LST0P = LST0P + 1
  302 PSCB = PSCB + 1
      IF(LST0P.GE.4) LWUSE = 0
      SKIPN9 = 1
      IF(LWUSE.EQ.1)WRITE(3,1887)
 1887 F0RMAT(' ')
      IF(PSLT.GE.1.AND.LWUSE.EQ.1)WRITE(3,1881)(STACKL(I),I=1,PSLT)
 1881 F0RMAT(10L2)
      IF(PSWT.GE.1.AND.LWUSE.EQ.1)WRITE(3,1880)(STACKW(I),I=1,PSWT)
 1880 F0RMAT('+',20X,1P20E20.5)
      IF(PSCB.GT.PSCT) G0 T0 390
  303 NUMB = STACKC(PSCB)
      IF(NUMB.LE.0) G0 T0 390
      IF(NUMB .EQ. 199) G0 T0 331
      IF(NUMB.LT.100) G0 T0 1
      NUMB = NUMB - 120
      IF(NUMB.LE.0) G0 T0 390
      IF(NUMB.GT.6) G0 T0 721
  304 G0 T0 (21,22,23,24,25,26,26),NUMB
  721 NUMB = NUMB - 10
      IF(NUMB.LE.0) G0 T0 390
```

```
      IF(NUMB.GT.9) GO TO 722
      GO TO (31,32,33,34,35,36,37,38,39),NUMB
  722 NUMB = NUMB - 10
      IF(NUMB.LE.0) GO TO 390
      IF(NUMB.GT.8) GO TO 723
      GO TO (390,390,390,390,45,46,47,48),NUMB
  723 NUMB = NUMB - 10
      IF(NUMB.LE.0) GO TO 390
      IF(NUMB.GT.9) GO TO 724
      GO TO (51,52,53,54,55,56,57,58,59),NUMB
  724 NUMB = NUMB - 10
      IF(NUMB.LE.0) GO TO 390
      IF(NUMB.GE.10) GO TO 307
      SKIPNO = NUMB
  712 SCOUNT = 0
  711 PSCB = PSCB + 1
      IF(STACKC(PSCB).NE.147) GO TO 711
      SCOUNT = SCOUNT + 1
      IF(SCOUNT.NE.SKIPNO) GO TO 711
      GO TO 302
C
  307 NUMB = NUMB - 10
      IF(NUMB.GT.9) GO TO 308
      N = NUMB
      LTOG10 = 1
      GO TO 302
  308 NUMB = NUMB - 10
      IF(NUMB.GT.9) GO TO 390
      L = NUMB
      GO TO 302
  390 WRITE(3,1090)
 1090 FORMAT(' CODE IN STACKC NOT ALLOWED')
      WRITE(6,1091)
 1091 FORMAT(' ILLEGAL CODE GENERATED, PLEASE CHECK INPUT STRING ')
      GO TO 601
```

```
C
C  INTERPRETING THE CØDES
C
C  EVALUATING THE ØPERANDS
C
    1 STACKW(PSWT+1) = Z(NUMB)
      STACKA(PSWT+1) = NUMB
      IF(NUMB.EQ.19) LFUNCT = 1
  351 PSWT = PSWT + 1
      GØ TØ 302
C
C  USING THE BINARY ØPERATØRS
C
   21 STACKW(PSWT-1) = STACKW(PSWT-1) +  STACKW(PSWT)
      GØ TØ 352
   22 STACKW(PSWT-1) = STACKW(PSWT-1) -  STACKW(PSWT)
      GØ TØ 352
   23 STACKW(PSWT-1) = STACKW(PSWT-1) /  STACKW(PSWT)
      GØ TØ 352
   24 STACKW(PSWT-1) = STACKW(PSWT-1) *  STACKW(PSWT)
      GØ TØ 352
   25 STACKW(PSWT-1) = STACKW(PSWT-1) ** STACKW(PSWT)
      GØ TØ 352
   26 VAL = 1.0
      LAST = STACKW(PSWT) + 0.5
      IF(LAST.EQ.0)GØ TØ 92
      DØ 91 I=1,LAST
      VAL = VAL*STACKW(PSWT-1)
   91 CØNTINUE
      STACKW(PSWT-1) = VAL
      IF(NUMB.EQ.7) STACKW(PSWT-1) = 1.0/VAL
      GØ TØ 352
   92 STACKW(PSWT-1) = 1.0
  352 PSWT = PSWT - 1
      GØ TØ 302
```

```
C
C   USING THE UNARY OPERATORS
C
    31 STACKW(PSWT) = SIN(STACKW(PSWT))
       GO TO 302
    32 STACKW(PSWT) = COS(STACKW(PSWT))
       GO TO 302
    33 STACKW(PSWT) = SIN(STACKW(PSWT))/COS(STACKW(PSWT))
       GO TO 302
    34 STACKW(PSWT) = ABS(STACKW(PSWT))
       GO TO 302
    35 STACKW(PSWT) = EXP(STACKW(PSWT))
       GO TO 302
    36 STACKW(PSWT) =  ALOG(STACKW(PSWT))
       GO TO 302
    37 STACKW(PSWT) = ALOG10(STACKW(PSWT))
       GO TO 302
    38 STACKW(PSWT)= INT(STACKW(PSWT))
       GO TO 302
    39 IF(STACKW(PSWT).LT.0.0)STACKW(PSWT) = -1.0
       IF(STACKW(PSWT).GE.0.0)STACKW(PSWT) =  1.0
       GO TO 302
    45 IF(.NOT. STACKL(1)) GO TO 330
       PSLT = 0
       GO TO 302
   46  STACKL(1) = .TRUE.
       GO TO 355
    47 IF(LTOG10.EQ.1) GO TO 360
       GO TO 302
    48 Z(STACKA(PSWT-1)) = STACKW(PSWT)
       PSWT = PSWT - 2
       GO TO 302
C
   330 CONTINUE
       PSLT = 0
```

```
      GØ TØ 712
  331 WRITE(6,2241)
 2241 FØRMAT('ODIFFERENTIAL EQUATIØN TØ USE NØT DEFINED,'
     *' PLEASE CØRRECT')
      GØ TØ 601
C
C  USING THE LØGICAL ØPERATØRS
C
   51 STACKL(PSLT-1) = STACKL(PSLT-1) .ØR. STACKL(PSLT)
      GØ TØ 354
   52 STACKL(PSLT-1) = STACKL(PSLT-1) .AND. STACKL(PSLT)
      GØ TØ 354
   53 STACKL(PSLT) = .NØT. STACKL(PSLT)
      GØ TØ 302
   54 STACKL(PSLT+1) = STACKW(PSWT-1) .GT. STACKW(PSWT)
      GØ TØ 353
   55 STACKL(PSLT+1) = STACKW(PSWT-1) .GE. STACKW(PSWT)
      GØ TØ 353
   56 STACKL(PSLT+1) = STACKW(PSWT-1) .LT. STACKW(PSWT)
      GØ TØ 353
   57 STACKL(PSLT+1) = STACKW(PSWT-1) .LE. STACKW(PSWT)
      GØ TØ 353
   58 STACKL(PSLT+1) = STACKW(PSWT-1) .EQ. STACKW(PSWT)
      GØ TØ 353
   59 STACKL(PSLT+1) = STACKW(PSWT-1) .NE. STACKW(PSWT)
  353 PSWT = PSWT -2
  355 PSLT = PSLT + 1
      GØ TØ 302
  354 PSLT = PSLT - 1
      GØ TØ 302
C
C  DEFINING THE DESIRED DIRIVATIVES
C
  360 DØ 391 I=1,9
      IF(I.GE.N) GØ TØ 361
```

```
      ZDOT(I) = Z(I+1)
  391 CONTINUE
      WRITE(6,1100)N
 1100 FORMAT(' N IS TO LARGE,N=',I5)
      GO TO 601
  361 ZDOT(N) = STACKW(1)
      LAST = N
      ZLARGE = 1.0D12
      DO 392 I = 1,LAST
      IF(ZDOT(I).GT.ZLARGE) ZDOT(I) = ZLARGE
      IF(ZDOT(I).LT.-ZLARGE) ZDOT(I) = -ZLARGE
  392 CONTINUE
      IF(LWUSE.EQ.1)WRITE(3,1110)(I,Z(I),I,ZDOT(I),I=1,LAST)
 1110 FORMAT(' Z(',I1,') =',F20.5,10X,
     *        'ZDOT(',I1,') =',1PE20.5)
      LSTAT = 0
      IF(M.GE.2) RETURN
      OUTPUT(3) DEL,T
      WRITE(3,8110)Z,ZT,ZDOT
      WRITE(3,8111)STACKC
      RETURN
  601 CONTINUE
      LSTAT = 3
      RETURN
      END
```

```
C
C   THE FOLLOWING SUBROUTINE NORMALIZES ALL DATA AS REQUIRED
C   BY SUBROUTINE RTOI
C
      SUBROUTINE NORM(NUM,X,Y,XN,YN,K)
      DIMENSION X(1),Y(1),XN(1),YN(1)
      DOUBLE PRECISION Z(10), ZDOT(10), DEL, T
      DIMENSION ZI(9)
      INTEGER STACKC(200)
      DIMENSION ZC(100)
      COMMON ZC,Z,ZDOT,DEL,T,A,B,C,D,E,F,G,H,
     *XSCALE,YSCALE,XCENA,YCENA,NXSIZE,NYSIZE,TI,TF,ZI,
     *XMIN,XMAX,YMIN,YMAX,XFACT,YFACT
      COMMON IDEV,NO,NN,LSTAT,N,STACKC,LFUNCT,L
      COMMON XCENT, YCENT
C
C   COMPUTING THE NORMALIZING (INCLUDES CENTERING) FACTORS
C   THIS IS BYPASSED AFTER THE FIRST CALL TO THE SUBROUTINE
C
      IF(K.EQ.1)GO TO 21
      XDIFF = XMAX - XMIN
      XFACT = NXSIZE/(5.*XDIFF)
      XCENT = XFACT*(XMAX + XMIN)/2.
      YDIFF = YMAX - YMIN
      YFACT = NYSIZE/(5.*YDIFF)
      YCENT = YFACT*(YMAX + YMIN)/2.
C
C   NORMALIZING THE X'S AND Y'S
C
   21 DO 91 I=1,NUM
      XN(I) = X(I)*XFACT - XCENT
      IF(ABS(XN(I)).GE.1.5) T = TF
      IF(XN(I).GT.1.5)XN(I)= 1.5
      IF(XN(I).LT.-1.5)XN(I)=-1.5
   91 CONTINUE
```

```
      DØ 92 I=1,NUM
      YN(I)=Y(I)*YFACT - YCENT -0.1
      IF(ABS(YN(I)).GE.1.5) T = TF
      IF(YN(I).GT.1.5)YN(I)= 1.5
      IF(YN(I).LT.-1.5)YN(I)=-1.5
   92 CØNTINUE
      RETURN
      END
```

BIBLIOGRAPHY

1. Burroughs B5500 Information Processing Systems, Reference Manual, Burroughs Corporation, 1964.

2. Cardenas, A. F., and Karplus, W. J., PDEL - A Language for Partial Differential Equations, Communications of the ACM, March 1970.

3. Clancy, J. J. and Fineberg, M. S., Digital Simulation Languages: A Critique and a Guide, 1965 Fall Joint Computer Conference, Spartan Books, 1965.

4. Elgerd, O. I., Control Systems Theory, McGraw-Hill Book Co., Inc., 1967.

5. Forsythe, A. I., and others, Computer Science: A First Course, John Wiley & Sons, Inc., 1969.

6. Gibson, J. E., Nonlinear Automatic Control, McGraw-Hill Book Co., Inc., 1963.

7. Higman, B., A Comparative Study of Programming Languages, American Elsevier Publishing Co., Inc., 1967.

8. Hopcroft, J. E. and Ullman, J. D., Formal Languages and their Relation to Automata, Addison-Wesley Publishing Co., Inc., 1967.

9. Hsiung, Y., Phase-Plane Methods, M. S. Thesis, Naval Postgraduate School, 1969.

10. MacMillan, R. H., Nonlinear Control Systems Analysis, Pergamon Press, 1962.

11. Rosen, S., Programming Systems and Languages, McGraw-Hill Book Co., Inc., 1967.

12. Stoker, J. J., Nonlinear Vibrations in Mechanical and Electrical Systems, Interscience Publishers, Inc., 1950.

13. Syn, W. M. and Linebarger, R. N., DSL/90 - A Digital Simulation Program for Continuous System Modeling, 1966 Spring Joint Computer Conference, Spartan Books, 1966.

14. Thaler, G. J., and Brown, R. G., Analysis and Design of Feedback Control Systems, McGraw-Hill Book Co., Inc., 1960.

15. Thaler, G. J., and Pastel, M. P., Analysis and Design of Nonlinear Feedback Control Systems, McGraw-Hill Book Co., Inc., 1962.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Professor George J. Thaler Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4.	Professor V. M. Powers Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
5.	Professor George A. Rahe Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	2
6.	LT Ronald D. DeLaura, USNR Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
7.	Dr. Toyomi Ohta Professor of Electronic Computation Defense Academy Obaradai, Yokosuka, Japan	1
8.	Mr. Wei-Mun Syn IBM Monterey & Cottle Roads San Jose, California 95114	1
9.	LT Harvey G. Nelson, USN 734 S. W. Austin St. Seattle, Washington 98106	2

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Postgraduate School Monterey, California 93940	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

A Compiler for the Interactive Solution of Differential Equations

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Electrical Engineer's Thesis; June 1971

5. AUTHOR(S) (First name, middle initial, last name)

Harvey Gordon Nelson

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1971	221	15

8a. CONTRACT OR GRANT NO. b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

A digital simulation language for the interactive definition and solution of piece-wise continuous non-linear ordinary differential equations using the Runge-Kutta method has been designed and implemented. The combination of interactive graphics approach and a special differential equation description language make the analysis program very versatile and easy to use. For second order systems, a grid of phase trajectory segments over the user specified phase plane is used as a background for the solutions.

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
ompiler						
iteractive Graphics						
ontrol Systems Analysis						
ontinuous Non-linear Differential Equations						

Thesis
N363
c.1
132612
Nelson
A compiler for the interactive solution of differential equations.